Synopsis for the title

"QoS in Health Care through IoT and Fog Computing"

submitted by

Mr. Kanani Pratik Bavchand Bhavna [FOTE/936]

as a partial fulfillment for

Ph.D in Computer Science and Engineering

at

The Maharaja Sayajirao University of Baroda

Research Guide

Dr. Mamta Chandraprakash Padole

Associate Professor

Department of Computer Science and Engineering

Faculty of Technology and Engineering

The Maharaja Sayajirao University of Baroda

Vadodara

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Health care is the maintenance or improvement of health via the prevention, diagnosis, treatment, recovery, or cure of disease, illness, injury, and other physical and mental impairments in people. While computer engineering enables the health care domain to record, process, convert and analyze the health care data. Other supporting technologies like IoT, Fog computing and Cloud computing can facilitate remote monitoring, remote medical assistance, local processing and in future storages of medical data.

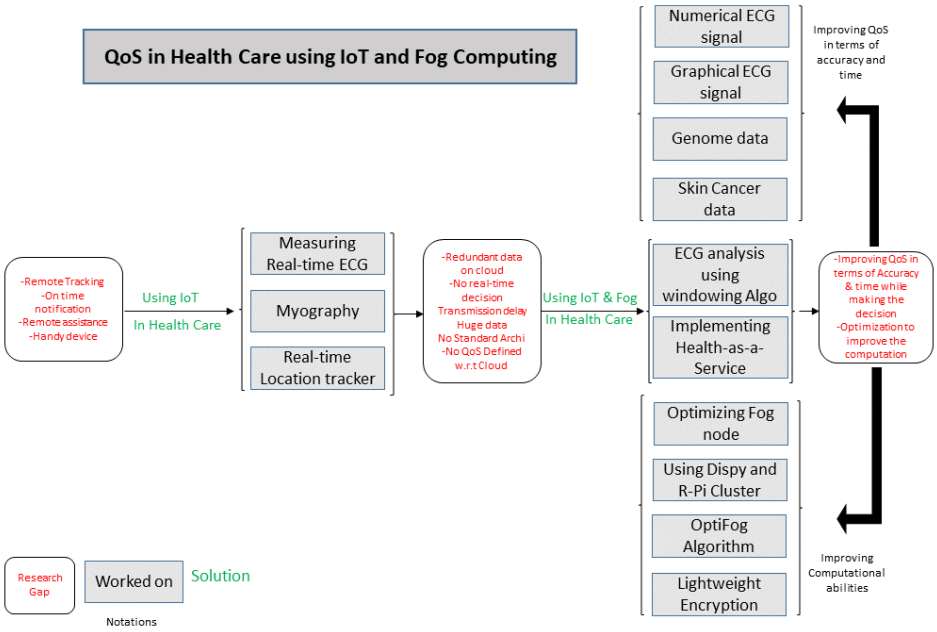
Internet of Things (IoT) is the network of objects or things which use hardware, software and communication protocols to support exchange of data with other entities over this network. IoT sensors are context sensitive but if rightly used then they are of great use in health care. In this report the IoT technology is used to record Real time ECG signals, Myography signals and GPS signals of the patient. The work is published in paper [1], [4] and [8] from the publication list.

IoT sensors can be used to remotely monitor a patient's health but these sensors usually generate large amounts of data which they are unable to process. In order to overcome this the data is sent over the cloud where the decision making takes place. But this transportation of data between the sensor and the cloud waste the network bandwidth and this is because in most health care cases the data is redundant and the transmission delays are also higher. Thus, the integration of IoT and cloud are not useful for making real time decisions. The solution to this problem is using the IoT along with Fog Computing. Fog Computing is the processing on the go computing, where the computation node is kept near the data source. It can be a local gateway or a device in the network having processing, storing and communication power. It is found that no standard Fog architecture and Quality of Service (QoS) parameters are mentioned to implement and measure the Health care services. By implementing real time ECG signal analysis using Raspberry Pi and windowing algorithm, we have proved that the Fog node is capable to do decision making in the health care domain. This work is published in paper [1]. Published paper [10] shows that the different QoS parameters and Health care architecture is defined for the fog computing to support health care domain.

The above implementations and experiments show that the fog computing node is able to process the data on real-time basis but it lags in terms of computation power which leads to delays and inaccuracies which are very undesirable in the healthcare field. In this report different health care signals like ECG signals, Skin cancer data and the Genome data are considered and their prediction accuracy is improved. In the ECG signal processing, the accuracy is improved for time based and image-based ECG signals using different augmentation techniques. In the Genome sequence, pattern matching is improved in terms of time by using different hardware and software techniques. Skin cancer data is also considered which is non real time data and the improvement is achieved in terms of prediction accuracy using machine learning techniques. The work is published in [2], [3], [5] and [7] paper.

To improve the computation power of the Fog nodes, different optimization techniques are used. The first technique used is called Operating System scheduling. This work is published in [6] paper. Further to boost the Fog node capability the distributed computing approach is introduced in the fog computing. To support this technique the raspberry pi cluster is used with dispy framework. And the results in [publication 9] confirm that the distributed computing in fog computing is efficient enough to provide faster executions to the real time data for real time decision making.

Further to make optimal use of available hardware different parameters are considered like number of cores, CPU usage, Main Memory and Response time of each device in the cluster. For every parameter the execution time is monitored and their weightage in the computation is determined. By using this weightage and worst-case scenario of algorithm analysis, the OptiFog algorithm is designed to get the optimization in the fog computing domain. The OptiFog algorithm is published in paper [11]. Different lightweight security algorithms and their background need is also studied. And based on that a Multi-level lightweight fog security algorithm is designed to give good security with better time and space complexity. In this report the major focus is given on Fog Computing, defining Fog computing and QoS parameters for Health care w.r.t Cloud. The other important milestone is OptiFog Algorithm and the Fog routing work. Chapter 2, chapter 3 and chapter 4 are discussing the same respectively. The scope of the above work is in the field of designing smart fog gateway, Optimizing fog node and in efficient fog routing.



Research Flow

1.1 Overview of Fog Computing

Fog Computing, popularly known as Edge computing is the novel paradigm/architecture that provides limited computing, storing and network services at the end user devices at user's edge. Simple fog computing architecture is shown in figure 1.

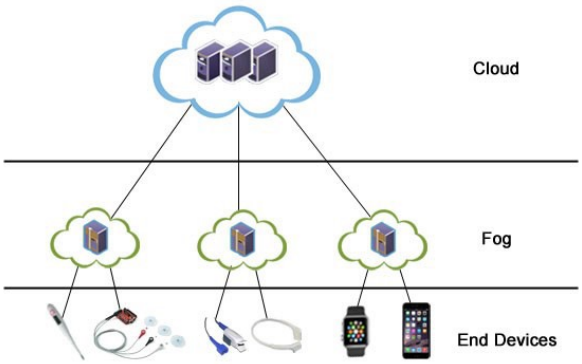


Fig. 1 Fog Computing Architecture

The implanted fog computing extends the cloud computing paradigm to the edge of end users and it is able to perform storage, processing and data forwarding. In the given figure the fog node/server is at layer-2. Whatever data gets forwarded by end devices will be first received by fog node. This node keeps on processing the data and is able to take actions and also it forwards the filtrated data to the cloud for future use and analysis. It has characteristics like

1) **Adjacent Physical Location:** Fog devices are near the user end, and hence they can process the user data with less delay and can be made custom in user's need context. They can use protocols of WLAN too.
2) **Support for on-line analytics:** Due to limited capability of processing and storing, Fog devices are connected to cloud servers where they can have on demand real time analytics of live data streams.
3) **Service is provided by smart but not powerful devices:** As the fog node has limited storing and processing power, it cannot do big data analytics because of its limited processing power and storage, but it can have real time decision making on certain conditions, when it occurs.
4) **Supports for various communication networks:** when different sensors connect to internet different protocols are involved. Some of the supported protocols are Bluetooth, Zigbee, WLAN, WiFi, 2g/3G/4G, WiMax and so on.
5) **Distributed computing:** Fog nodes can communicate to other fogs. And cloud sees the environment as a collection of different fogs, i.e., the whole computation needed by user is available via distributed fogs.

1.2 Fog Computing in Healthcare domain

In recent years, the dramatic growth of Internet of Things (IoT) linearly increases the unprecedented volume and variety of stream data. IoT [1] is a dynamic and global network infrastructure interconnecting objects with unique identities for diverse advanced application services. Despite offering the advanced services, IoT is incapable in processing and storing the massive amount of data due to its limited storage and processing capacity. Cloud computing technology has unlimited capabilities regarding processing and storage resources, resolving the inconvenience of IoT by providing the virtual resources in pay-as-you-go manner [2]. Although, the vast availability of cloud computing resources, services and applications, but several kinds of such resources are not completely attained due to the latency concerns.

Owing to the rapid increase in internet-connected smart devices and several service requests, a heavy burden comes to the network bandwidth and degrades the Quality of Service (QoS). Also, the high network latency between the smart devices and the cloud is infeasible for delay-sensitive applications [3]. Fog computing [4] is the most promising paradigm that significantly reduces the latency and provides the advantages of cloud computing by extending the cloud resources to the network edge [5]. It offers distributed services and allows the knowledge generation and data analytics of the streams generated by the smart IoT devices. The benefits of fog computing are especially useful for pervasive healthcare monitoring applications [6]. The IoT plays a crucial role in constantly monitoring the physiological status of the hospitalized patients without the need of actively engaging the caretaker [7]. Healthcare monitoring applications [8] widely rely on the Wireless Body Area Networks (WBAN) which is the most underlying technology in healthcare IoT. WBAN assists in ubiquitously acquiring the physiological information involving Electrocardiography (ECG), Electromyography (EMG), blood pressure, glucose level sensing, Blood Pressure monitoring, Oxygen sensing, rehabilitation, medication and blood temperature in the efficient and unobtrusive way. To effectively support the pervasive healthcare applications, the prior research work utilizes the

cloud computing technology for IoT devices [9]. The conventional fog computing methods [10] present variety of solutions by focusing on the different application scenarios in mitigating the service latency. However, these techniques are still in its infancy stage in attempting to provide the services to pervasive healthcare computing in real-world. Thus, the proposed approach focuses on introducing a smart fog gateway by applying the smart partitioning and decision-making using the linear decision tree in fog environment and optimally utilizing the cloud resources for the healthcare IoT requests. It intends to reduce the response latency and increase the resource utilization, which is defined in SLAs while providing the service to healthcare applications. Fog Computing Architecture [17] in health care domain is as follows:

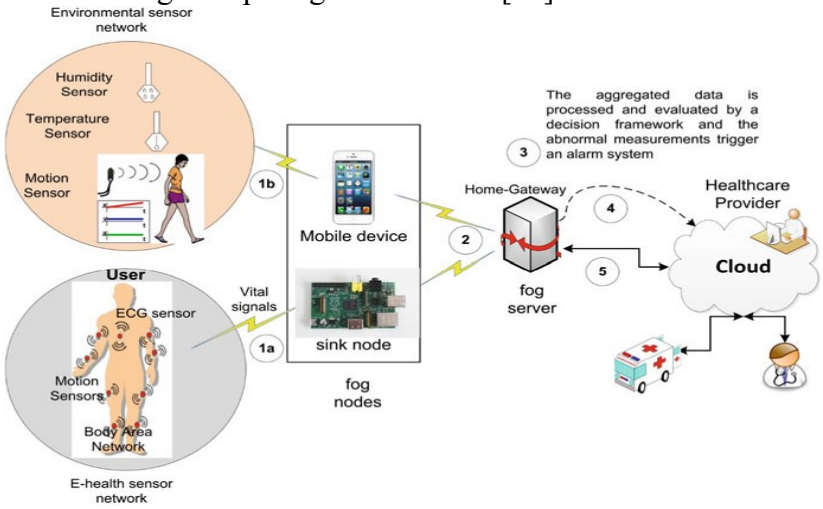


Fig. 2 The use case of daily monitoring for provisioning timely healthcare services with low latency

1.3 Research Outcomes

- To extend the cloud computing to the fog computing to support latency-sensitive healthcare IoT applications
- To design the smart fog gateway with smart allocation to satisfy the Service Level Agreements (SLAs) in terms of ensuring the optimal response time and resource utilization
- To develop an algorithm to dynamically take a decision regarding IoT stream health care data and decision-making in fog and cloud
- Interactive communication by fog layer even if connectivity to the cloud layer is not present
- Low latency health care application for delay-sensitive real-time applications
- To develop an algorithm, to dynamically decide node capability, that enables to identify the node that can be assigned the appropriate task for health care applications in Fog Computing
- Improved QoS in terms of prediction accuracy and Computing Optimization in Fog Computing
- Introduced Distributed computed in Fog Computing
- Efficient routing algorithm in vehicular fog computing infrastructure

Chapter 2: Fog computing as a smart Gateway in Health Care

When IoT, Fog and Cloud Computing are combined, the resultant system's performance is far better than the performance of the individual systems. Hence, the combination results in a very efficient Health Care system. Fog and Cloud Computing have their dimensions that not only support each other but also explore many new application domains. In this paper, the real-time ECG based Health Care system is considered and implemented with Cloud and Fog Computing. Different QoS parameters like memory consumption, transmission delays, computation delays, network delays, CO_2 emission, data transferred and response time are measured, analyzed and improved to make the system more efficient. And based on the Fog computing characteristics and capabilities, the Raspberry Pi 3 B+ model is configured as a Health Care serving gateway by using different installation and configuration steps. The proposed system is subjected to multiple ECG streams for varying numbers of patients to find its limitations. Every QoS parameter is explored in detail for decision-making time. In the end, the Fog computing based proposed system is concluded for its pros and cons and future aspects of the Fog node are discussed to make the system better.

2.1 Literature survey

Table 1 shows the reviewed references in tabular form. It highlights the goal of research, technology used and the current challenges/future improvements.

Table 1. Future challenges/improvements in literature review

Reference	Work goal and technology used	Challenge/improvement in terms of
[1]	Use of IoT in Health care	QoS in Traffic
[2]	Integrating IoT and Cloud, using different cloud computing services	Pervasive Health care system generates • Vast amount of data • Needs Intelligence in IoT
[3]	Using IoT and developing wearable Body area network	• Context aware cloud architecture • Data privacy and visualization
[4]	Studies advantages of Fog computing in IoT	Resource management in Fog node
[5]	Integrating edge and Cloud computing	Needs application development strategies that is where to deploy application instance, failure recovery, edge node security and distributed bandwidth management
[6]	Duties of Fog node in Health care	• Local storing and computing implementation in fog • Use of fog in Body area network (BAN)
[7]	IoT Health care	• Standardization • IoT health care platforms • Cost • Scalability • QoS • Data protection
[8]	IoT in Health care	Integration of runtime sensing information into health care records
[9]	IoT and Cloud computing in pervasive Health care	Realization of Health care through mobile devices has challenges like • Storage and management of data • Security and privacy of data • Ubiquitous access

[10]	Survey and analysis of Fog Computing	• Reliability • Networking • Capacity • Security • Application aware provisioning
[11]	Low latency support communication in cloud computing	• Enable communication through NAT, firewalls • Finding and solving connectivity issues automatically
[12]	IoT in Health care	• Security aspect • PAMIoT with more devices (scalability)
[13]	IoT and Cloud	• Resource allocation based on application context • QoS model in IoT and Cloud environment
[14]	An overview of Fog computing and IoT	• Fog user interface • Fog interface designing to manage IoT data and Cloud
[15]	Fog micro datacenter and data filtering	• Smart Fog gateway • Extend model based on cloud service customers
[16]	IoT and Ubiquitous computing	Impact of this model on health care domain applications
[17]	IoT and Fog Computing In Health care	Upgrading fog nodes in terms of its capabilities to provide other possible services
[18]	Fog computing and IoT	• Providing QoS in Fog Computing with more heterogeneous services • Device mobility factor
[19]	Cloud of things	Effect on QoS after trimming and pre-processing the data for heterogeneous applications
[20]	IoT and Fog computing in Health care	• Large number of hardware components are used • Limitations are not tested • In ECG wave analysis only P and T waves are used
[21]	IoT and Local processing	Not tried and tested on real time data to make decisions in health care systems
[22]	IoT and Fog computing	• Simulation based • SLA-aware flow placements and resource scheduling
[23]	IoT, Cloud computing and Fog computing	• Can be considered for health care applications • Simulation based
[24]	Fog computing and IoT	• Complex hardware • Cloud is not involved for future data reference • Multiple protocols are used for communications • Decision time in not compared with other systems/benchmarks
[25]	Fog computing in health care	• Only idea is discussed • Design and develop real time Fog based medical data analysis
[26]	Providing edge intelligence in Fog computing	• Mobile phones are used as edge devices • Used multiple detectors • Results are not compared with other systems

2.2 Problem statement

Nowadays, IoT-driven healthcare applications play a vital role in the distributed environment. IoT constantly generates the massive amount of stream data, which leads the complexity in handling this huge amount of data streams in the IoT devices itself. Since, the IoT devices are resource-constrained devices with the limited storage and processing capability, especially network resources. Moreover, the integrated cloud and IoT technology also impose several challenges for the end-users, network, and the terminals associated with the problem of high congestion, fast battery consumption, and low scalability. Since, long distance between the smart IoT devices and the cloud server creates the gap in providing the

response, which leads to latency issue. Accordingly, the latency issue creates a greater negative impact on the healthcare applications as healthcare applications are the delay-sensitive applications in real-world. Even though fog computing paradigm provides the opportunities to the end-users, Cloud-Fog interface encompasses several challenges such as context-based resource allocation, workload imbalance, and service overhead. It consumes more time to identify the available VMs from the distributed fog environment to centralized cloud environment, which degrades the performance the service when dealing with the delay-sensitive healthcare applications. Hence, there is an essential need of satisfying QoS, ensuring quick response time and better resource utilization. Moreover, Fog computing does not have the ability to perform the compute-intensive process, to provide the massive storage, and to establish the wide area connectivity. Also, dividing the computing of the application in the fog and sending the compute-intensive process to the centralized cloud is arduous task due to the occurrence of high network latency. Thus, this work targets on providing the smart fog gateway for delay-sensitive healthcare applications by smart partitioning and allocation. i.e, fog node receives continuous stream of the healthcare data, first it needs to break the data stream into the chunks of data. Further these data chunks are processed by Fog in order to take actions like whether to forward this data to cloud or to process the data chunk by decision tree which will take an action on real-time basis.

2.3 Research gap

Most of the formerly presented fog computing research work presents the different architecture and framework for latency reduction in healthcare IoT systems. However, these methods are not able to effectively utilize the fog as well as cloud environments with the knowledge of application context and the resource availability. The resource management in the distributed fog environment is the challenging task. Nonetheless, the prior fog gateway architectures fail in manipulating the task orchestration and partitioning the IoT data streams from the massive amount of stream data. Moreover, it lacks in optimally allocating the application of the edge devices and storing the context-defined fog computing data in the transient fog storage.

2.4 Aim and Objectives

- To extend the cloud computing to the fog computing to support latency-sensitive healthcare IoT applications
- To design the smart fog gateway with smart allocation to satisfy the QoS in terms of ensuring the optimal response time and resource utilization
- To develop an algorithm to dynamically take a decision regarding stream and process execution of IoT application in fog and cloud.

2.5 Proposed methodology and system architecture

In the proposed methodology, a real-time health care system is taken into consideration. The system consists of a Data source, Fog Node, Gateway, Decision making, and messaging service. The real-time ECG signal is acquired from the patient [30]. These signals are recorded and sent to Cloud as well as the Fog Computing node. The windowing algorithm [31] is used to find the reference points PQRST in the ECG Signal. Based on these points the ECG time intervals are found out. Later decision making is done to find whether the given ECG signal is normal or abnormal.

In the proposed architecture shown in figure 3, the real-time ECG signals are sent to the Cloud and Fog node simultaneously. The task of performing analysis is carried out on both the

systems on the same signal. The generated results are then compared in terms of different parameters like computation time, transmission time, CO_2 generated and the total response time.

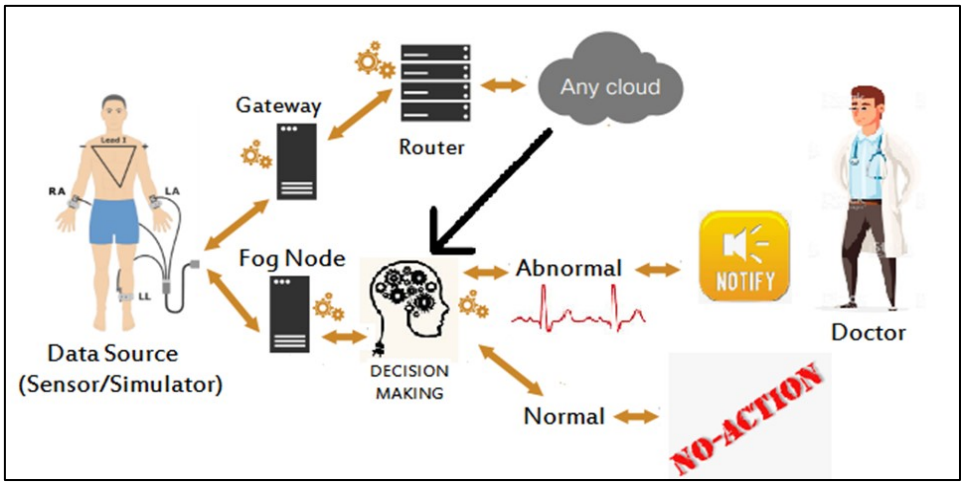


Fig. 3 Cloud computing and Fog Computing based Health Care System

Whenever an abnormality in the signal is found, at that very instant a text message consisting of the timestamp, signal interval values, and the patient data, is sent to the doctor and based on the timestamp the response efficiency is calculated.

2.6 QoS Parameters

Different QoS parameters like memory [32-33], Transmission delay, Computation delay, CO_2 emission measurement [34-37], data transferred and Response time are as follows.

2.6.1) Memory: The amount of memory utilized by a programming module can be calculated using Java methods. One can use the Runtime class functions for finding the memory utilized.

```
long usedMemory = Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory();
```

2.6.2) Transmission Delay: Transmission delay is the total time taken by the network to send the data from one point (source) to the second point (Destination).

2.6.3) Computation Delay: Computation Delay is the total time taken for the computation. In the proposed system it is the time measure after the entire signal is received in the system until it is processed and the output as normal or abnormal is produced. Based on the computation time the Speed up in the Fog Computing can be calculated by

$$[\![Speedup]\!]\_overall = [\![Execution\ time]\!]\_cloud / [\![Execution\ time]\!]\_fog$$

2.6.4) CO_2 Measurement: Fog Computing uses far fewer resources than the Cloud Computing infrastructures which reduce the CO_2 generation. The amount of CO_2 that can be saved from Fog Computing is further explained below**.** In 2011, According to Cisco 1.8 ZB of data was sent to the Cloud data centers. So, if 5.12 kWh of energy is required to send 1 GB data across then the total energy required to send 1.8 trillion GB of data is 9.216 trillion kWh of energy. To generate this much amount of energy, a total of 5.76 trillion kg of CO_2 is emitted. Using the above inferences, to transfer 1 Byte of data 2.98 x 10^{-3}mgm of CO_2 is emitted.

Now, if one can adapt to Fog Computing and assume that the data only travels to the data center only for storage purposes, then one reduces the CO_2 emission by 50%, and the CO_2 emission can be potentially reduced by 2.88 billion metric tons. That is almost 34 times more savings when compared to Cloud Computing. The amount of CO_2 can also be found out by finding the power used by the devices and relating CO_2 with power.

$$Power_{static} = Current_{static} * Voltage$$

$$Power_{dynamic} = \frac{1}{2} * Capacitive\ load * Voltage^2 * Frequency\ switched$$

2.6.5) Data Transferred: The data transferred is measured in bytes. Here, the ECG signals are sent to the Cloud and the Fog node. The network distance is measured in terms of the number of Hops. Usually, the Fog Computing node is at least 2 to 3 hops away from the data source.

2.6.6) Response Time: For the proposed system architecture, the Response time considered is the overall response time of the ECG signal processing unit. Here it shows the total time span starting from the ECG signal generation until the final response is generated and given to the doctor. It includes the addition of other delay times like Processing Delay, Queuing Delay, Transmission Delay, and the Propagation Delay.

$$d_{response} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

For the current system, the timestamp value of the Java programming language counts as the current system time in milliseconds which is noted by using the java method System.currentTimeMillis(). Here, the final system improvement of the Fog computing is shown by the difference of Response time taken by both the systems.

2.7 Defining different timestamps to measure the QoS

In this system, different parameters are defined to measure the QoS parameters, and arithmetic relations among these parameters are used to calculate different delays. The timestamp Parameters with their denotations are as follows.

Table 2. Different Time stamp calculations

Different Timestamp instants	QoS Parameters
$₮_{gen}$ = ECG signal generation time	Overall Response Time improved by Fog = $₮_{pc} – ₮_{pf}$
$₮_{rf}$ = Time at which ECG signal is reaching the Fog Node	Transmission Delay (Fog) = $₮_{rf}$ - $₮_{gen}$
	Transmission Delay (Cloud) = $₮_{rc}$ - $₮_{gen}$
$₮_{pf}$ = Time at which ECG signal is processed at Fog Node	Fog Computation Time = $₮_{pf}$ - $₮_{rf}$
	Cloud Computation Time = $₮_{pc}$ - $₮_{rc}$
$₮_{rc}$ = Time at which ECG signal reaches the Cloud Computing	End to end Data Transferred = Data bytes x no. of Hops
	CO_2 generated = Data Transferred x emitted CO_2/Byte
$₮_{pc}$ = Time at which ECG signal is processed by the Cloud Computing	

2.8 Experimental Results

The web interface is developed to understand the ECG signal processing and its analysis in more detail. The same web interface is run on both Cloud and the Fog Computing node. The interface shows the "Patient Name", a red color label if signal Abnormality exists, the ".txt" file name where the ECG signals are stored for processing, different QoS parameters, ECG waveform, patient details, and different intervals for each ECG wave along with their abnormality. If the signal is found abnormal then the system will send SMS to the Health Care supporting staff.

Table 3. QoS parameters and their values

Parameters	Cloud Computing	Fog Computing	Improvement
Transmission Delay (ms)	7677	117	7560
Computational Delay (ms)	55	670	-615
Data Transferred (bytes)	35982	11994	23888
CO_2 Emitted (mgm)	107.22	35.74	71.48
Response Time in ms (Time Format in Java)	1547121433052	1547121426107	6945

The Fog and Cloud computing system is first subjected only for 1 patient, and all resultant parameters are measured and shown in Table 3. The system shows that Fog computing surpasses overall response time performance than Cloud computing and performs better in terms of Response time, data transferred and CO_2 generated. But Fog computing is hanging back in terms of computation power. Cloud computing takes only 55 ms to computer the given job while the Fog processor takes 670 ms. System is tested against by varying the number of patients to study the system behavior in depth. Each and every parameter value is taken and shown as an average value for n patients to discuss further.

2.8.1) Transmission Delay: Transmission Delay depends on many factors like the number of hops between the source and the destination, available network bandwidth, layer conversion, VPN set up, wired-wireless configurations, congestions, tunneling, and the number of users, etc. The proposed system is tested for a different number of patients from 1-5. And the average transmission time is shown in figure 4.

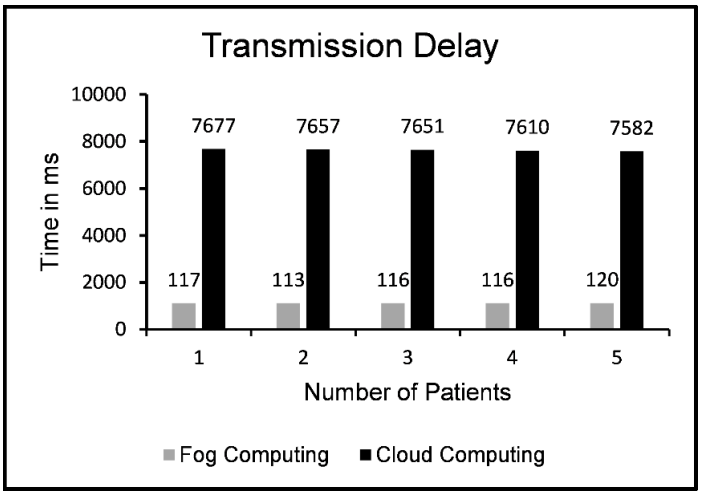


Fig. 4 Transmission Delays of Fog and Cloud Computing

Figure 4 makes it clear that the transmission delay in terms of Cloud and Fog Computing remains almost the same for the same source and destination for a different number of patients. And it is also evident that the transmission delay in Cloud computing is very much higher than the Fog computing.

2.8.2) Computation Delay: Computing power depends on the device capability in terms of its hardware configuration. It depends on cache memory, processor, operating frequency, scheduling algorithm, communication bus, memory and number of cores. The current Fog device is configured to read and analyze more real-time ECG waves simultaneously on its different ports. By varying the number of patients its average computational delay in fog node is measured and shown in figure 5.

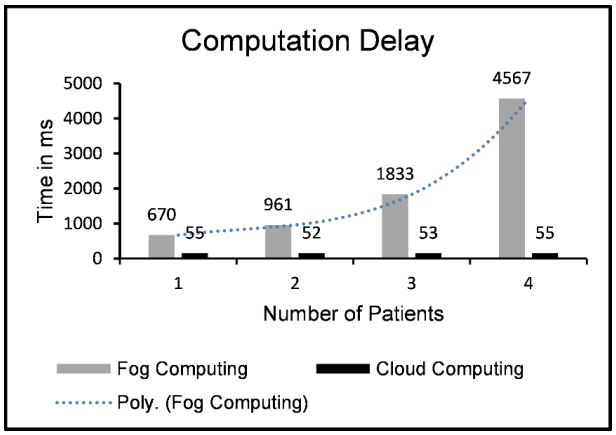


Fig. 5 Computation delay of Fog and Cloud computing

The computation delay is almost identical in terms of Cloud computing but it varies a lot in the case of Fog computing. Fog computing shows the polynomial growth of the order of 3˚. The computational delay also depends on how the system is made i.e. the GUI computation, background computation, the refresh rate and the number of parallel tasks, etc. For four patients the computational delays are 4523, 4662, 4487 and 4593 ms respectively, which comes out as 4567 as an average value shown in figure 5. So giving more load on the Fog system is not suitable for the time-sensitive decision-making systems.

2.8.3) Response Time: The Response time is the overall performance time of the system. It shows the time difference between the generation of the ECG signal and the generation of response or decision in terms of normality and abnormality. Here, the average response time is found by varying the number of patients as shown in figure 6.

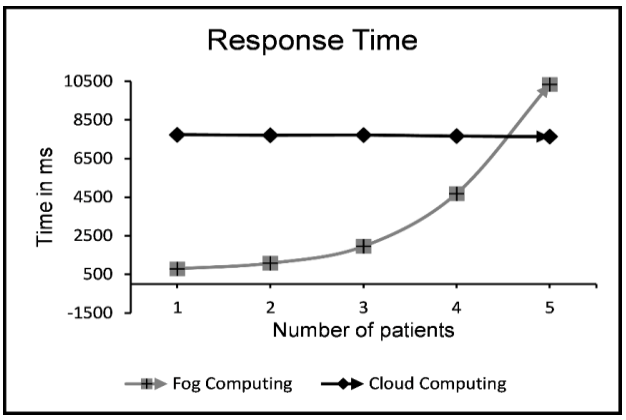


Fig. 6 Response time of Fog and Cloud Computing

In the proposed system, the Fog node serves early responses than the cloud architecture, but only when the number of users is lesser than four. It responds very early if the number of users is less. While in case of Cloud computing the Response time is almost equal. If the number of users is five or more than that then the proposed Fog computing architecture will underperform than the cloud. Hence it is not suggested for more number of patients in real-time health care analytics.

Fog computing gives a better response when the number of patients is less and it is observed that the Fog computing response time is directly proportional to the number of patients. After a particular threshold value of the number of patients, fog computing will not perform better than Cloud computing.

Chapter 3:

OptiFog: Optimization of Heterogeneous Fog Computing for QoS in Health Care

3.1 Introduction

A patient's life can be saved if it is possible to make quicker decisions based on faster processing of real-time health care data, such as ECG processing. To achieve faster decision making, contemporary health care applications use cloud computing for such data. When cloud computing is used, data transmission deferrals may cause delays in the decision-making process. To overcome this, Fog Computing is used. Fog computing saves energy, bandwidth and prevents transmission latencies but, lacks in computing power as compared to Cloud Computing. To enhance the computing power of the Fog node, a Cluster of Raspberry Pi having heterogeneous configurations can be used. In Health Care applications the Fog Computing performance can be assessed by measuring the time elapsed between the generation of the health care data and decision-making. In this paper, ECG signal analysis is taken as a processing job in Fog Computing. *Dispy* is used to facilitate the scalability and parallel data processing on a Cluster of Raspberry Pi used for Fog Computing, to enable faster decision making. Further, the performance of the Raspberry Pi cluster-systems using dispy are analyzed and optimized step by step based on different parameters. The first parameter is data transmission time which is improvised by minimizing network overheads. Other optimization parameters like CPU usage, number of cores, response time and available memory space, these parameters are considered and varied, to assess the performance of Heterogeneous Raspberry Pi cluster. Based on the results obtained, a novel optimization approach "*OptiFog*" is proposed to achieve faster computation in worst-case scenarios by varying and assigning jobs to the nodes to measure performance parameters in Distributed Fog Computing. "*OptiFog"* assures a minimum 10% improvement in the performance of the Fog Computing environment.

3.2 Aim and Objectives

- To extend the distributed computing in the fog computing to support latency-sensitive healthcare IoT applications
- To design the smart fog computing cluster with smart job allocation to satisfy the SLAs in terms of ensuring the optimal response time and resource utilization
- To develop an algorithm, to dynamically decide node health to assign the appropriate task for health care applications in fog computing
- To develop an optimized performance model in the health care domain to serve the community

3.3 Focus of the chapter

- Considering a Low latency health care application for delay-sensitive real-time applications
- Analyzing dispy performance in fog-distributed computing
- Analyzing and understanding other processing factors and techniques to get the best out of it
- Explaining different performing parameters and their impact on the computation
- Developing the algorithm best suited for health care applications and Raspberry pi cluster in Fog computing

3.4 Literature Review

The literature review studied is targeting the systems which have used or optimized system related to Distributed computing in terms of performance. It also focuses on the usage and improvements in the system due to memory, response time, CPU usage and the number of cores. Later it is aiming at the work related to fog computing in context to performance improvement.

Helen Karatza et al. [52] have addressed the issues faced while scheduling parallel jobs on a cluster of distributed processors. Two types of routing schemes are considered. Also, 3 types of scheduling techniques are considered. The objective of this paper is to analyze the performance of these task scheduling algorithms in each case of routing. Various other system parameters that need to be considered for job submission and task scheduling are also studied and tabulated in this paper. This paper analyses the feasibility of each scheduling algorithm in each routing scenario and the impact of the system parameters in task scheduling. Simulation is used to analyze the performance of the algorithms in different system load conditions. By analyzing the simulation we can understand the impact of the scheduling policies on the system performance.

Abdou Guermouche et al. [53] have proposed a system that aims at improving the working of a parallel multifrontal solver, MUMPS. This scheduling approach is memory-based. Memory constraints are used to choose a processor's slaves and/or associates. Slaves are chosen according to their memory availability. It aims at reducing the used stack size at run time. Li Xiao et al [54] have proposed a paper that tries to enhance the effective usage of global memory. Job distribution strategies are also built accordingly. When a node has insufficient memory to accept jobs, the extra load is then migrated to other associates with sufficient memory availability. Unbalanced memory allocations for jobs cause page faults, so the motive is to minimize the same thereby improving efficiency. The load sharing policy proposed improves the performance of memory-bound jobs. Yuyan Sun et al. [55] have proposed a paper in which they describe distributed systems. In distributed systems, it is essential to have a fair load-sharing policy to ensure that the computational capacity is completely utilized. The load sharing policy has a major impact on performance. The system memory has a major role in system performance. Therefore available memory becomes the base of the load sharing systems. In memory-bound jobs, memory-based load sharing system has higher performance. The developed algorithm shows better performance than FCFS and Round Robin algorithm in load sharing systems. The memory-based load sharing systems are more adaptive in terms of performance and they are sensitive towards the memory variance. Kizhakkethil et al. [56] present a memory-based hybrid Dragonfly algorithm for optimization.

Nawwaf Kharma et al. [57] have proposed H2GS which is a two-phase scheduling algorithm. It works on distributed systems, while the main focus is on heterogeneous systems. A highly efficient schedule is generated using a heuristic list-based algorithm that makes up phase one. In phase two shorter schedules are evolved. Tasks to be scheduled are given priorities. The ready task with the highest priority is selected for scheduling. The next phase is where the processor is selected. Here a task is selected and submitted to the processor to minimize the time of execution. In the paper presented by Haluk Topcuoglu et al. [58], an algorithm for scheduling is implemented on processors whose numbers are predefined. The motive is to meet efficient scheduling and enhanced performance simultaneously. The name of the algorithm is Heterogeneous Earliest Finish Time (HEFT). At each step, the maximum upward rank value is chosen and assigned to a processor. Based on an insertion-based approach, the earliest completion time is minimized. HEFT algorithm is robust and performs well over a wide range of graph structures.

Bao Liu et al. [59] have presented several scheduling/co-scheduling techniques employed in distributed systems. Predictive scheduling and Proportional-sharing scheduling are the types of local scheduling introduced here. Predictive scheduling provides adaptivity, intelligence, and proactivity to adopt new architectures and changes in the environment automatically. It learns new architectures, algorithms, and methods that are embedded in the system. The allocator aggregates previous inputs, in the form of a vector of performance information (CPU usage), into sets. Each set corresponds to a scheduling decision. Sets are split or merged, to keep a limited memory demand, by the allocator. Marjan Khosravi Talebi et al. [60] have presented an algorithm that is used for scheduling in the cloud computing environment. In this algorithm, parameters like processor status are used to obtain the node on which the job should be scheduled. The goal is to obtain an efficient scheduling method that minimizes the overall processing time of all the loads by distributing the loads amongst all the available processors. The processor to which the current job has to be assigned is decided by a formula that takes into account the history of scheduling along with the processing power and link time.

Alfredo Goldman et al. [61] have presented a review article about the different scheduling algorithms used in distributed computing in cloud computing. The prominent difference between distributed computing and cloud-based computing is the incorporation of virtualized systems. Virtual Machine (VM) allocation is performed to strengthen the server. Here, the scheduler pool is denominated as the software being considered for scheduling. Hardware requirements (number of cores in the system and their usage statistics, etc.) are used by the scheduler for scheduling. Zafeirios Papazachos et al. [62] have studied various gang scheduling algorithms and analyzed their efficiency for clusters consisting of multi-core systems. Gangs are scheduled in multi-core cluster systems using the suggested migration structure. An evaluation model provides results on the performance of the system. In gang scheduling, fragmentation is caused when the size of the gangs prevents them from fitting in idle cores. Flexible and adjustable schedules are made by dynamically migrating parallel jobs. Migrations are given importance as it satisfies the requirement of load balancing.

Salim Bitam et al. [63] have focused to develop a job scheduling task for mobile users in Fog computing. They have used the Bees Swarm algorithm with CPU execution time and the total amount of memory. One of the important aspects is to save the network bandwidth in communication, Frank et al. [64] have suggested to compress the raw data and resend it, and decompress it for processing on the receiver side. But, this compression and decompression save the network bandwidth but increases response time in health care which is very critical.

3.5 Understanding the Working Environment

To implement the distributed computing for Raspberry Pi cluster in fog computing “dispy” is selected. Dispy is developed in python and python works very well with Raspbian operating systems. The main problem faced in deployments of distributed computing is that each slave node has to be configured for a particular application context. If more slaves add-up then it needs more configuration. This makes scalability in distributed computing a bit difficult, but in case of dispy only master node has to be configured and on all slave nodes, only dispy should be installed. No need to configure every slave. This makes a distributed system more scalable if we use dispy.

To perform the experimentation different hardware and software configurations are chosen. Here, we will understand the need for different configurations with their specifications. Each node in the system is preloaded by some computation and the processing job with its details are discussed here.

3.5.1 The need for heterogeneous configuration

In this system, two setups are used. In both the setups we are varying the number of nodes in the cluster from 1 to 4. This is to understand the effect of computation-distribution by techniques like memory, response time, CPU usage and the number of cores. And we have taken two types of nodes in the system and maximum nodes are four in the cluster. In one setup all 4 nodes are homogeneous in terms of its hardware. But, in another setup three homogeneous and one heterogeneous node is taken. This is to see the effect of processing, load distribution, overall performance effect and adaptiveness of the algorithm in a cluster, in different configurations and parameters.

3.5.2 Hardware Configurations of the cluster nodes

For the projected system, Raspberry Pi 3 model b+ [67] and Raspberry Pi 4 [68] models are used. The hardware configurations are as follows.

Table 4. Hardware configurations of cluster nodes

Hardware Module	Raspberry Pi 3 Model b+	Raspberry Pi 4
Processor	Broadcom BCM2837B0, Quad-core Cortex-A53 (ARMv8) 64-bit SoC	Broadcom 2711, Quad-core Cortex-A72 64-bit SoC
Operating Frequency	1.4 GHz	1.5 GHz
Bluetooth	4.2	5.0
Wi-Fi	2.4 GHZ / 5.0 GHZ IEEE 802.11.b/g/n/ac/wireless LAN	2.4 GHZ / 5.0 GHZ IEEE 802.11.b/g/n/ac/wireless LAN
Memory	1GB LPDDR2 SDRAM	4GB LPDDR4 SDRAM
SD Card Support	Micro SD card	Micro SD card
Operating voltage and current	DC 5V/2.5A DC	DC 5V/3A

The Raspberry Pi 4 node is having a higher hardware configuration than the Raspberry Pi 3 b+ model. The Pi 4 node is higher in terms of processing, communication, and memory. Purposely the higher node is introduced in the system so that the system behavior can be studied over the other parameters and a good algorithm can be designed accordingly. And in all varying nodes, the dispy server node is always Raspberry Pi 3 b+. The details are given below in table 5.

Table 5. Number of nodes and nodes selection

Experimental Configuration	Number of Raspberry Pi 3	Number of Raspberry Pi 4
1 node	1	-
2 nodes	2	-
3 nodes	3	-
4 nodes	4	-
4* nodes	3	1

3.6 Continuous Load on cluster nodes

To design the best suitable algorithm to process health care data efficiently in the Fog clustering environment. All cluster nodes are kept busy in some or the other computation work apart from health care data processing. This computation keeps cluster node occupied till a certain level of CPU usage and with that, the node is allowed to perform the health care data processing task. To create the system load, the "stress" tool is used in the Raspbian operating system environment. The “Stress” is a tool [69] used to test system performances when they are loaded. System admin uses this tool to see the performance of I/O syncs, VM status, cache thrashing, CPU usage, driver performance, and process creation and termination. This tool can generate different sorts of load on the system as specified in the option field. And it can

continue generating that much stress on the system till the said time ends. Here, the following command is used to generate the CPU load for the needed time.

The "**sudo stress –cpu 1 –timeout 20000**" command which keeps CPU busy for nearly 25% for 20000 seconds, till we run and test all algorithms in the system.

3.7 Processing Job Description and logic

ECG is a periodic wave [70], it repeats its cycles after a certain interval of time. It has P-Q-R-S-T-U points as their reference points representing respective peaks. Now the important part is to get PR, QRS and QT intervals out of these waves for each and every wave.

Table 6. Standard ECG Intervals for a healthy adult with standard bpm

Intervals	Normal Value	Normal Variation
QT Intervals	400 ms	±40ms
QRS Interval	100ms	±20ms
PR Interval	160 ms	±40ms

And based on the given table, one can find the time intervals in milliseconds and by comparing it with table 6 [71-74] we can find whether ECG waves are normal or abnormal. The normal beats per minute (bpm) is 60 to 100 bpm. Further, the windowing algorithm [75] is used to detect different intervals. R peaks are prominent peaks in the ECG waves, and here they are the highest values in the cycle. After detecting different intervals and comparing it with table 7, the wave is normal or abnormal is discovered.

3.8 Executing ECG waves using a dispy manner

The series of ECG waves are divided into a set of two waves to find the ECG intervals. From these ECG intervals, the prediction for the wave is normal or abnormal is made. Here, each job given by master dispy node to the slaves containing two ECG waves and the process continues for a different number of waves. The time taken by different nodes for a different number of waves is as shown below in figure 7.

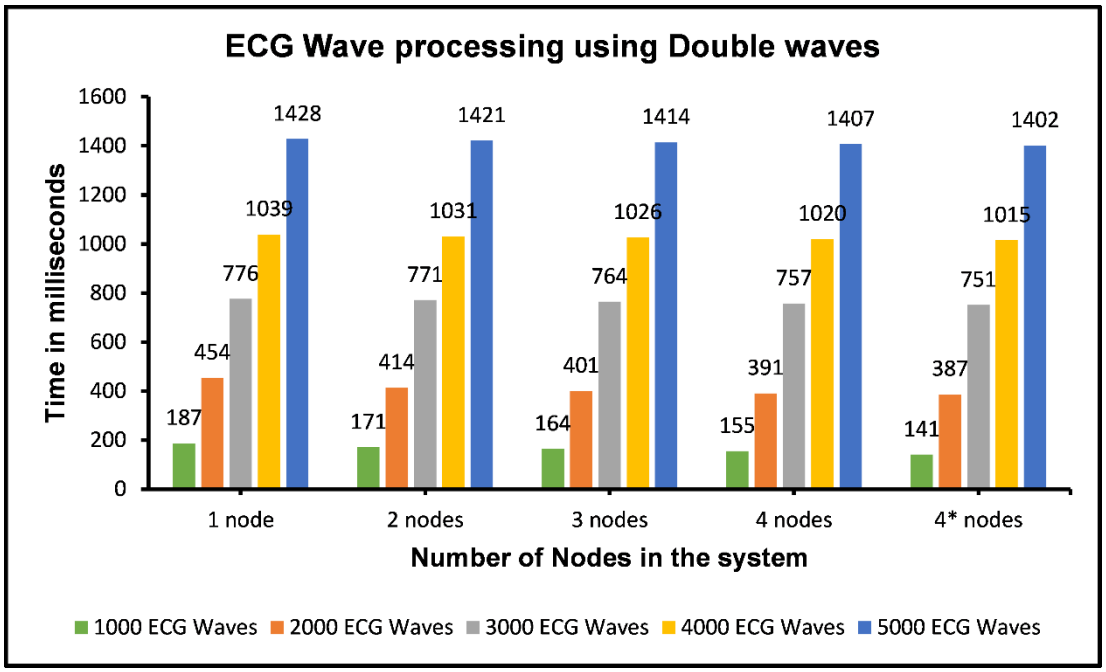


Fig. 7 ECG wave processing using two ECG waves in one sub job

For the system set up mentioned used to get figure 7, the dispy master node have to do so many iterations to complete the job. So to reduce the number of iterations varying number of waves are tried every time in the dispy job. And obtained results are taking lesser time than the current two waves in the job. The Tetra, Octa and Deca waves are tried. The performance of octa is higher than double and tetra waves. But the deca waves are not performing better than the octa waves. Hence, octa wave is taken as a reference for further improvement.

3.9 System symbols, functions and performance indicators and their implications

The different functions, terms, and symbols defined in the proposed system are listed below.

A. System parameters

Table 7. System symbols and description

Symbol	Description
J	Complete available job at the master node
J_i	Sub job of J, $J_i \subseteq \{J\}$: sub job with i ECG waves
M	Master Node
S	Slave Node
S_i	i^{th} Slave Node
n	Number of available slave nodes in the dispy system
X_i	X^{th} factor for i^{th} slave node
s	Sub job in sending context
r	Sub job in receiving context
T_s	Sending time of a particular sub-job on the master node
T_r	Receiving time of a particular sub-job on the master node
T_{is}	Sub job sending time to i^{th} node from master node
T_{ir}	Sub job receiving time of i^{th} node on the master node
T_{ijk}	T is timestamp value, where i: is slave node number, j: is the sub-job number, k ϵ {s,r}
N	Total number of sub-jobs
J_{ci}	Result of completed sub-job by i^{th} node
$SelectNode_i$	i^{th} node is selected for the further set of sub-job processing
RT	Response time
Ŋ	Ŋ ϵ {normal, abnormal}, i.e., the characteristic of ECG wave
CU	CPU usage
MU	Memory usage
NC	Number of cores in the cluster node
∀	For all
RT_o	sum of all RT_i , i∀ {1,2,…n}
MU_o	sum of all MU_i , i∀ {1,2,…n}
O_i	string_object having values timestamp, MU_i, CU_i, and NC_i
O_{ijr}	i: is slave number, j: is the sub-job number and r: is receiving context
O_r	Returning object by i^{th} slave
Ᵽ	Priority factor
Ȼ	Capacity factor
Ł	Time factor
μ	Memory factor
ψ	Impact factor
$\boldsymbol{\alpha}$	Number of jobs in one go, where each job has set of eight ECG waves
$\boldsymbol{\alpha}_c$	Result of completed $\boldsymbol{\alpha}$ jobs
$J_{\boldsymbol{\alpha}}$	Sub jobs with $\boldsymbol{\alpha}$ number of jobs
*	Multiplication
ms	Milliseconds
P_{RT}	Performance time taken as Response time
P_{CU}	Performance time taken by CPU usage technique

P_{MU}	Performance time taken by memory usage technique
P_{NU}	Performance time taken by a number of cores technique
$P_{OptiFog}$	Performance time taken by OptiFog algorithm
slave_id	Number representing a slave
job_id	Number representing a sub job

B. System Functions

Table 8. System functions and description

Functions	Description
timestamp sendTask(sub job, slave_id);	sends sub-job data to a particular slave bearing the mentioned id and returns the timestamp of that event
timestamp receiveTask(result, slave_id);	receive the result of the given task from the slave having the id number and note the timestamp of that event
timestamp getTimestamp( );	returns timestamp
CPUUsageQueryCU(slave_id);	returns CPU usage of slave_id
MUsageQueryMU(slave_id);	returns memory usage of slave_id
NC QueryNC(slave_id);	returns number of free cores of slave_id
timestamp getFactors(subjob, slave_id);	returns timestamp when giving sub job to slave_id
string_objectreceiveFactors(job_id, slave_id);	returns string_object after processing job_id on slave_id
calculate(Ȼ, Ł, µ);	calculates capacity, time and memory factor

C. Finding different Optimization parameters and using them in the system to understand its effect on computation

To find different parameters and to introduce optimization in the system, different computation parameters are studied. Every parameter have their significance in computing systems. These parameters are Response time, CPU usage, number of cores and memory.

3.10 OptiFog Algorithm

The proposed idea is intended to perform optimally in the heterogeneous scenario, by exploiting the most available processing power present in the system. We have used four techniques that are memory-based, Response-time-based, CPU-usage-based and number-of-cores-based. Every technique is run when every node was busy in some other computation work. This is just to find out which technique is more weighted and less weighted in heterogeneous computing scenarios. And when the obtained graphs and results are analyzed for greater jobs and higher nodes, it is found that the CPU usage results are the best and the second is the number of cores. Response time gives the third greatest performance followed by memory technique.

3.10.1 OptiFog Algorithm and its insights

OptiFog is a hybrid optimization algorithm that finds the impact factor based on all the above four mentioned techniques. This impact factor is the value of every node and based on this value, the number of jobs are allocated to each node in one go. Every node will submit the job and its current status of CPU, Cores and memory to the master node. Master nodes compute the Response time and impact factor for each node in every iteration and based on the impact factor (ψ) value, it will assign the number of jobs.

The CPU and cores on every node have different capacities based on Operating Frequency, processor specifications, cache size and the bus size. It represents the processing capabilities of a node. That is the other main reason to give higher priority (Ᵽ) to this factor. Whereas the memory and response time of a particular node can be compared with other nodes in terms of

size and unit like GB and ms. Therefore, these two units memory and response time are seen as collective units in the distributed system.

Impact factor is the overall health status of a node in terms of memory, CPU, cores and response time. But OptiFog uses three main factors to find the impact factor. That is Capacity (Ȼ), Memory (μ) and Time (Ł).

a) Capacity Factor (Ȼ): this factor is based on the CPU usage and number of cores technique. The CPU usage is related to the number of cores. And these both techniques are giving a very good performance which is almost similar. So in this case, these two values are combined and the factor is calculated as

$$Ȼ = NC * (1 - \mathrm{CU}), \text{where CU } \epsilon [0,1]$$

b) Memory Factor (μ): In this factor, if MU_i is less than the node performs better. Thus, every node MU_i is found out and scaled to 1.The factor is calculated as

$$\mu = \frac{(1 - MU_i)}{\sum_{i=1}^{n} MU_i}$$

c) Time Factor (Ł): The response time of a node is inversely proportional to its capacity. By keeping this in mind the factor is designed in such a way that the node with high response time will get low rank and the node with less response time will be treated with high ranks.

$$Ł = \frac{\sum_{i=1}^{n} RT_i}{RT_i}$$

After finding Ȼ, μ and Ł. The final ψis calculated as

$$\psi = 3Ȼ + 2Ł + 1\mu$$

where,numericals ϵ {Ᵽ}

3.10.2 The OptiFog Algorithm is as follows

Algorithm 1 OptiFog Algorithm to process the Job J

Input: Complete job J having continuous ECG waves,

Output: Ƞ for each and every wave

1. **procedure** OptiFog
2. initialize RT_o =0, MU_o = 0
3. T_s←getTimeStamp();
4. **for** i← 1, 2, ……. n **do**
5. T_{ijs}←getFactors(J_i , S_i);
6. O_{ijr}←receiveFactors(Jc_i , S_i);
   T_{ijr}← get(O_{ijr});
   MU_i← get(O_{ijr});
   CU_i← get(O_{ijr});
   NC_i← get(O_{ijr});
7. RT_i←T_{ijs} - T_{ijr}
8. RT_o =RT_o + RT_i
9. MU_o =MU_o + MU_i
10. **end for**
11. $\boldsymbol{\alpha}_i$← 1, ∀ i ϵ [1,2,…n].

12. $\psi_i \leftarrow 0, \forall\ i \in [1,2,\ldots n]$.
13. j = n+1, i=1
14. **while j<=N do**
15. calculate($\mathfrak{C}_i$, $Ł_i$, μ_i);
16. $\psi_{old} \leftarrow \psi_i$
17. $\psi_i = 3\mathfrak{C}_i + 2Ł_i + 1\mu_i$
18. **if** $\psi_i > \psi_{old}$ **then**
19. $\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_i + 1$;
20. assignTask($\boldsymbol{\alpha}_i$, i);
21. **end if**
22. **if** $\psi_i < \psi_{old}$ **then**
23. $\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_i - 1$;
24. assignTask($\boldsymbol{\alpha}_i$, i);
25. **end if**
26. **if** $\boldsymbol{\alpha}_l >= 0$ **then**
27. $j = j + \boldsymbol{\alpha}_i$
28. **end if**
29. i = i + 1
30. **if** i > n **then**
31. i = 1
32. **end if**
33. **end while**
34. $T_r \leftarrow$ getTimeStamp();
35. $P_{OptiFog} \leftarrow T_{ir} - T_{is}$
36. **end procedure**
37. **Procedure** assignTask($\boldsymbol{\alpha}$, i)
38. **if** $\alpha_i > 0$ **then**
39. $T_{is} \leftarrow$ getFactors($J_{\boldsymbol{\alpha}}$, S_i);
40. $O_{ir} \leftarrow$ receiveFactors($J\boldsymbol{\alpha}c$, S_i);
$T_{ir} \leftarrow$ get(O_{ijr});
$MU_i \leftarrow$ get(O_{ijr});
$CU_i \leftarrow$ get(O_{ijr});
$NC_i \leftarrow$ get(O_{ijr});
$RT_i \leftarrow T_{ir} - T_{is}$
41 **end if**
42.. **end procedure**

After running the OptiFog Algorithm, the obtained results are shown in figure 8 below.

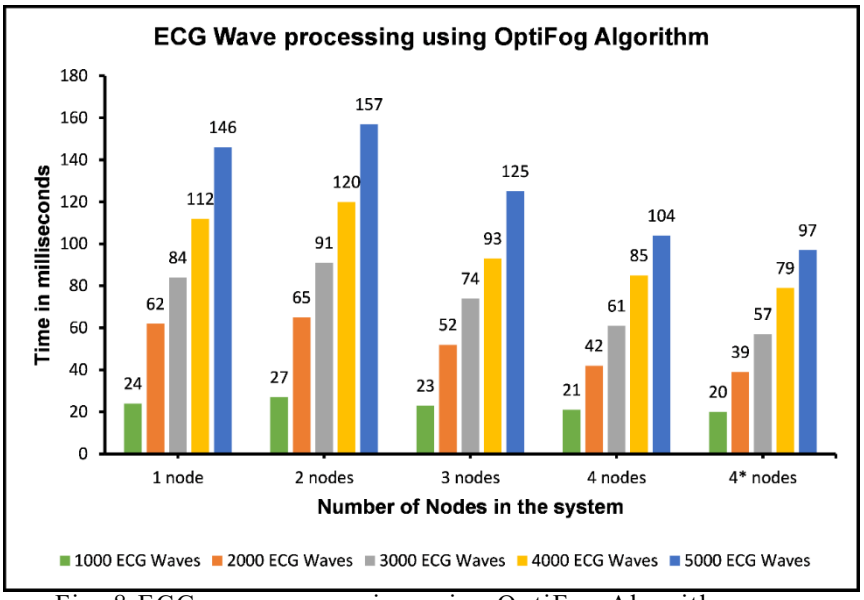


Fig. 8 ECG wave processing using OptiFog Algorithm

Graph Interpretation

- Effect of increasing nodes can be seen
- 4* nodes give very good performance results
- It takes less computation time than ECG octa wave and other techniques

3.11 OptiFog Algorithm Testing

The OptiFog algorithm is tested in three ways to prove its rationality. In the first test case, the speedup factor is considered. In the second test case, the algorithm is run for a higher number of ECG waves to see its performance. And in the last test case the OptiFog algorithm is run for dispy Deca wave case where dispy system which was not giving good performance due to pre-loaded nodes in the cluster.

3.11.1 Test case 1: Speedup

Based on the above experiments done so far in the Raspberry Pi clustering environment, every parameter or the technique is indicating the way to improvise the performance. These improvements are reducing the job time J, which can be compared by using the Speedup factor to see its impact.

$$Speedup_{overall} = \frac{Execution\ Time_{old}}{Execution\ Time_{new}}$$

Here, the performance time of the dispy technique with double ECG waves is taken as a benchmark time and other techniques are compared to it. Also, the step by step speedup between other techniques is also shown in Table 9.

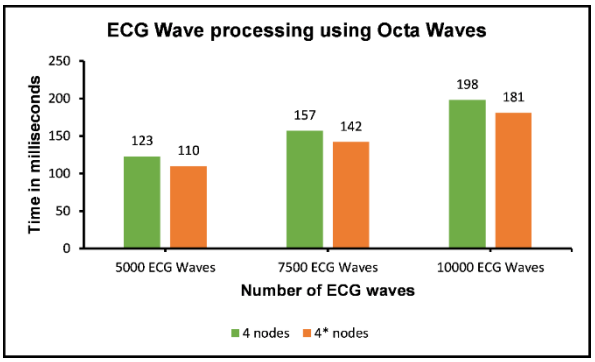
Table 9. Speedup factors for 5000 ECG waves for 4 nodes and 4* Nodes system

Technique used (q)	Speedup w.r.t Benchmark		Speedup w.r.t (q-1) technique	
	4 Nodes system	4* Nodes system	4 Nodes system	4* Nodes system
Double waves	1	1	-	-
Tetra Waves	5.9118	5.8417	5.9118	5.8417
Octa Waves	11.256	12.5178	1.904	2.1428
OptiFog Algorithm	13.529	14.4536	1.2019	1.1546

Table 9, shows that the series of algorithms are considered are improving the performance. The OptiFog gives the maximum speedup of 14.4536 in 4* Nodes system and it speeds up the performance by 1.1546 with respect to Octa waves in 4* Nodes system.

3.11.2 Test Case 2

The proposed algorithm OptiFog is tested on a greater number of ECG waves. The system is tested against the normal dispy system with octa waves and OptiFog algorithm. The system is kept under a loaded scenario to see the performance of the OptiFog algorithm under the worst-case scenario. The number of ECG waves are taken as 5000, 7500 and 10000. The results obtained by the dispy system and OptiFog algorithm is shown in figure 9 and 10.



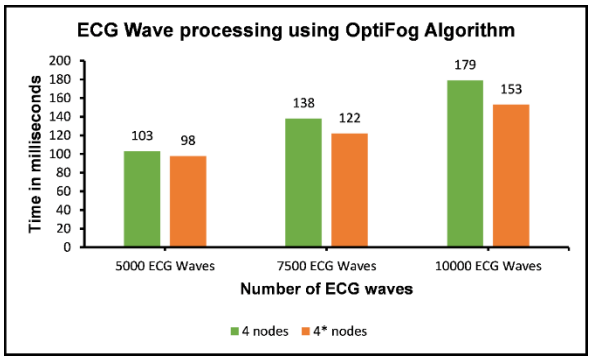


Fig. 9 ECG wave processing by dispy system, Fig. 10 ECG wave processing by OptiFog Algo.

From Figures 9 and 10, it is observed that OptiFog outperforms the dispy system and other algorithms in terms of computation. OptiFog shows better and better results for larger jobs. In this test case, the effect of 4* nodes in terms of performance is very noticeable. For 4* Nodes system OptiFog is showing a speedup of 1.183 for 10000 ECG waves. This factor was 1.1546 for 5000 waves. This confirms that the Speedup factor is improved for higher number of waves. The overall percentage of improvement is shown in figure 11. As the health care processing load increases the OptiFog performs better and its computational performance is increases concerning dispy system.

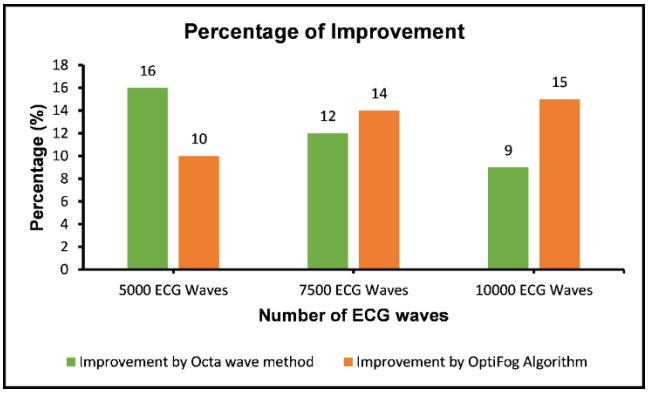


Fig. 11 Percentage of improvement in the given test case

3.11.3 Test case 3

In this test case, the system where dispy was processing ECG waves using deca waves is considered. And when doing this the performance was dropping. In this case, the network overhead is observed lesser because a number of waves are deca, but the loaded nodes are not

able to handle the deca wave loads and they are taking more time than expected. But when the same case is considered using OptiFog Algorithm which predicts the job size based on individual node health status using impact ψ value. The result of test case 3 is shown below where OptiFog algorithm is run using deca waves.

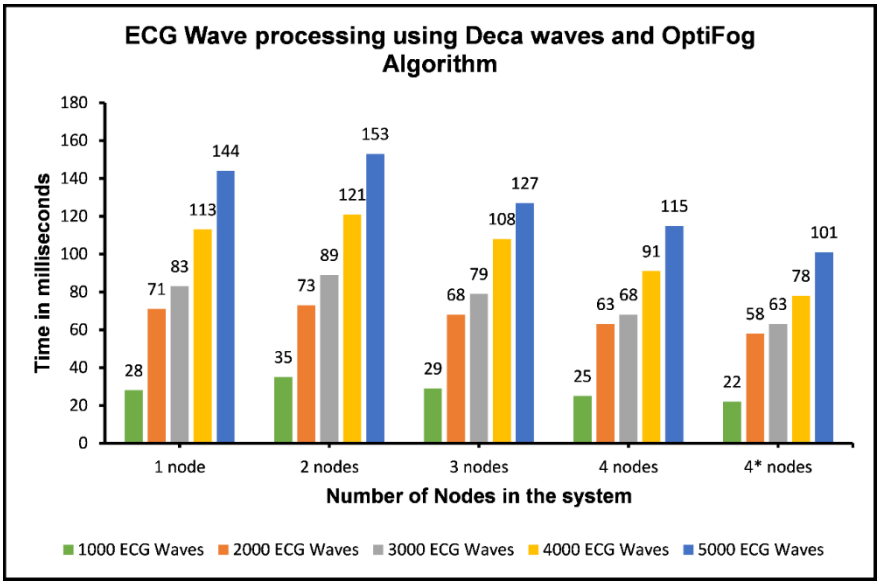


Fig. 12 Running Deca waves using OptiFog Algorithm

Graph Interpretation

- No improvement is seen for 1 node system in comparison to the dispy deca wave graph because of job load and existing preload
- In 2 Nodes system, the performance is degrading because of the calculation overhead of Ϲ, Ł, μ and ψ.
- For 3 Nodes system, the results are better than 1 Node and 2 Nodes system as now Job assigning and job size variation starts for available nodes
- 4 Nodes system performs better than 1 Node, 2 Nodes and 3 Nodes system
- Finally, 4* Nodes system outperforms because OptiFog is able to detect good impact factors every time and able to assign more and more task in sub job for 4* system

OptiFog uses Distributed computing to strengthen itself, and it is real-time processing algorithm expected in [76].

Chapter 4:

Lightweight Multi-level authentication scheme for Multi-level IoT-Fog Context

A multitude of cryptographic algorithms can be used for IoT devices' security, but because it provides limited resources like limited memory, insubstantial RAM, limited power, it becomes difficult to use such cryptographic algorithms intensively [78]. Thus, there is a need for a lightweight cryptography algorithm that can offer security efficiently to IoT devices without taking up a significant load. Effective implementation of IoT demands low latency and small chip area occupied on hardware or memory requirements for the software execution. Lightweight encryption supports both these requirements, and thus it compliments IoT, making it the best choice of security algorithm for IoT.

The proposed scenario illustrates the need for fog nodes at different levels and how they communicate with one another with the help of horizontal and vertical communication. Real-life scenarios are shown to demonstrate how lightweight encryption algorithms could be used at different layers to ensure that the security is preserved. Having multiple security algorithms make it more difficult to deploy any application, while the high complexity because of the multiple security schemes makes the system more vulnerable. So here, one unique lightweight security scheme is suggested which can single-handedly satisfy all the levels of security with different security levels. The scheme is based on Logical operations and it is multi-keyed logic.

The proposed scheme makes sure that it has a lower time and space complexity to support IoT devices. In addition to that, the scheme has varied key-length feature which makes it capable to secure higher-level data at higher-level devices like servers. In designing, different HMACs, Op_codes, and chaining logics are used. The proposed algorithm is also compared with other existing algorithms and it is discovered to be superior to the others. Furthermore, it is tested against various attacks to prevent any vulnerabilities from the exploitations. The given security scheme is lightweight and it provides security assurance in data transmission in the IoT-Fog, Fog to multi Fog, and Fog to Cloud end.

Further experiments are carried out to find the time and space requirement of the proposed algorithm with other existing algorithms. Few existing algorithms are chosen and they are executed on Apple MacBook Air devices. The time taken in ms by different algorithms are as shown in figure 13.

In terms of memory requirement, the Lightweight proposed scheme is taking the minimum time. The other algorithms are executed for a 256-bit key length whereas the proposed scheme is executed for 1024 bit key lengths. The time taken by the proposed scheme is very less because it uses logical operation to get avalanche effects in the ciphertext whereas most other algorithms are using multiple rounds of permutations and combinations with bit shifting mechanism.

The memory requirements of different algorithms are found. The comparison is as shown in figure 13. The memory requirement of the proposed scheme is higher than the TEA and BlowFish algorithm. It is because the existing scheme generates multiple keys to use the chaining mechanism and to assure the role of key serializability in the encryption and decryption process. The total memory taken by the proposed algorithm for 1024 bit key length is around 4 MB, which is very much feasible in the lower-level devices. To get it more feasible in much lower level devices the key length can be reduced further.

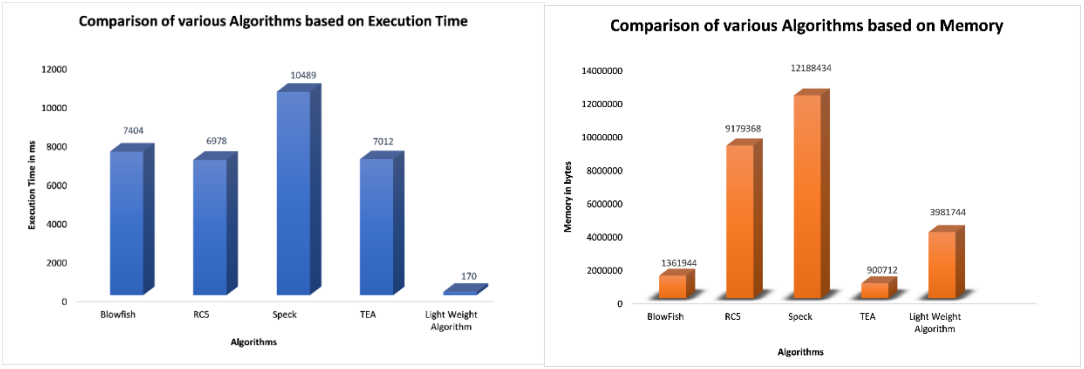


Fig. 13 Time and space complexity of Lightweight security algorithm

The time and memory taken by the suggested algorithm shows that the current scheme is more efficient than the existing algorithms. The initial key sharing in the proposed algorithm is designed in such a way that the initial handshaking in the communication process always remains secure. It also satisfies all three principles of security that is confidentiality by encryption, integrity by using hash functions, and availability by using Op_codes and sequence of ordered keys. Hence the proposed scheme is light-weighted and more secured, and is applicable in IoT-Fog-Cloud communication scenarios.

Chapter 5:

Other optimizations and developments made in Health Care domain

To upkeep Health Care domain, different existing systems are chosen and optimized in terms of performance. Also, different systems are developed based on real time signal analysis. The work done is classified as enhancements and developments. It is described in section 5.1 and 5.2.

5.1 Different enhancements to existing systems

Different existing health care related systems are studied and improvements are done to achieve higher accuracies. These systems are as follows.

5.1.1 ECG Image Classification using Deep Learning Approach

Cardiovascular Diseases are a major cause of death worldwide. Cardiologists detect Arrhythmias i.e. Abnormal Heart Beat with the help of an ECG graph which serves as an important tool to recognize and detect any erratic heart activity along with important insights like skipping a beat, a flutter in a wave and a fast beat. The proposed methodology does ECG Arrhythmias Classification by CNN, trained on grayscale images of R-R interval of ECG signals. Outputs are strictly in the terms of a label that classify the beat as normal or abnormal with which abnormality. For training purpose, around one lakh ECG signals are plotted for different categories and out of these signal images, noisy signal images are removed, then Deep Learning Model is trained. An image-based classification is done which makes the ECG Arrhythmia system independent of recording device types and sampling frequency. A novel idea is proposed that helps cardiologists worldwide, although a lot of improvements can be done which would foster a "wearable ECG Arrhythmia Detection device" and can be used by a common man. Proposed system aims to help cardiologists worldwide and with the developments of AI in the health care sector, our study will add value in this domain. This work proposes to help refine the vast clinical data, find patterns amongst them and to improve the accuracy parameter which is of paramount importance to serve a patient well. By implementing classification using Deep Learning, certain Machine Learning tasks like feature extraction and noise filtering have been avoided. Using the intense computational power to learn from data, workable accuracy has been achieved. The proposed model is achieving the accuracy level of 97.78 %. This chapter does not wish to supplant the brilliance of cardiologists and their expertise but through our tool wish to add value to their work. So, this chapter is to bridge the gap between technology and expertise in a health monitoring system in our case-Cardiology.

5.1.2 Deep Learning to Detect Skin Cancer using Google Colab

Deep Learning can detect features through self-training models and is able to give better results compared to using Artificial Intelligence or Machine Learning. It uses different functions like ReLU, Gradient Descend and Optimizers, which makes it the best thing available so far. To efficiently apply such optimizers, one should have the knowledge of mathematical computations and convolutions running behind the layers. It also uses different pooling layers to get the features. But these Modern Approaches need high level of computation which requires CPU and GPUs. In case, if, such high computational power, if hardware is not available then one can use Google Colaboratory framework. The Deep Learning Approach is proven to improve the skin cancer detection as demonstrated in this paper. The model also aims to provide the circumstantial knowledge to the reader of various practices mentioned

above. The model is trained for 100 epochs and the model achieves 77.98% test accuracy and 77.31% validation accuracy and approximately 82% training accuracy. Which is better as compared to the results depicted in the existing model which has an accuracy of 77.03%.

5.1.3 ECG Heartbeat Arrhythmia Classification Using Time-Series Augmented Signals and Deep Learning Approach

Electrocardiogram (ECG) signals are the best way to monitor the functionality and health of the cardiovascular system and also identify ailments related to it. Abnormal heartbeats are reflected in the ECG pattern and such abnormal signals are called as Arrhythmias. Automated classification and identification of the ECG arrhythmia signal that provides faster and more accurate result is increasingly becoming the need of the moment. Various machine learning skills have been applied to advance the accuracy of results and increase the speed and robustness of the models. A lot of focus has been given to the architectures and datasets employed but preprocessing of the data being equally important. In this, a preprocessing technique that significantly improves the accuracy of the deep learning models used for ECG classification is proposed with a modified deep learning architecture that adds to the training stability. With this preprocessing technique and deep learning model, the system is able to attain accuracy levels of more than 99% without overfitting the model. Applying augmentations to the dataset can not only make the model training more accurate but also stabilize it at higher accuracies. The proposed model consists of 6 residual blocks which means there is scope of overfitting the data but the augmented dataset also prevents overfitting by making classification difficult in the testing phase. The proposed model still displays high accuracy in such conditions. Thereby depicting its caliber to make highly accurate predictions with an accuracy rate of 99.12%.

5.1.4 Improving Pattern matching performance in Genome sequences using Run Length Encoding in Distributed Raspberry Pi Clustering Environment

Genomics and bioinformatics have grown as an independent field and are an area of active research currently. In this work, we first discuss the idea of the genome, its importance and wide-ranging applications in healthcare and medicine. We then give a brief overview of the various file formats used presently to store and manipulate genomic data along with their benefits and shortcomings. Moving on, we analyze and elaborate on how parallel and distributed computing can help in processing large files with genomic data and more importantly address the problem of pattern matching. In particular, we introduce the raspberry-pi and discuss in detail how a Raspberry Pi can accomplish the pattern matching task and how a Raspberry-Pi cluster can enable parallel computing to improvise the performance. This work also includes a brief account of various high-level application programming interfaces and libraries that can be used to parse genomic data like BioPython and Fuzzywuzzy. We conclude this work by summarizing some techniques that can be used for encoding and compressing files with genomic data like Run-Length encoding and genome differential compression respectively among other methods. The proposed work improvises the Genome pattern matching time by more than 50% in a distributed Raspberry Pi Computing environment. This system is useful to detect a particular genome-based abnormality where different pieces of Genome records are present on the different nodes in distributed computing and the input is given to find the disease. It can detect the abnormality within a short time. Also the different approach used here is to use Raspberry Pi based clustering to save cost and the energy. Due to the vast data size of even a single genome and its DNA information, there was an implicit need to find various ways in which the formats of these genome data file is easily accessible, like VCF, FASTQ, etc. This research is a stepping stone, for the usage of various techniques and

utilization of different available resources like Raspberry Pi in some highly extensive programming languages and their algorithms to help facilitate the problems faced for Genome Pattern Matching and its sequencing. In this section, only three Raspberry Pi are used and the increase in performance is very high. To increase it furthermore R-PI can be deployed. This concludes that by using compression techniques and distributed computing approach, the Genome pattern matching efficiency can be increased. In future, such systems can be used where each node is having the entire cancer genome sub files and the master node sends the pattern to each node, to detect the cancer.

5.1.5 Exploring and Optimizing the Fog Computing in Different Dimensions

Fog Computing facilitates very fast and secured response by saving network transmission to cloud and hence delays. In this paper, Fog Computing is explored in terms of its architecture, characteristics, functioning and purpose. Several fog nodes which are available in the market are studied and represented which can be used for Fog Computing developments. Existing applications and different Fog based applications are discussed along with the pros and cons of Fog Computing. Different optimization techniques to improve the Fog computing are discussed in detail with their impact on performance time. Such optimization techniques help to make health care application more efficient in terms of response time. Here, Fog Computing domain is explored in possible dimensions and relevant enhancements are discussed. Fog computing finds its application in various time sensitive and network saving applications by providing high mobility and security. Fog node can work in more than one context in the given scenario. Its performance can be optimized by using various optimization techniques. The technique using higher priority compare to other top five processes performs better than other mentioned techniques. It gives good optimization in terms of time complexity. Thus, fog computing is the noble choice for delay sensitive decision-making applications.

5.2 Different system developments to facilitate Health care domain

Sensors help to record real time signals and IoT sends this signal to the processing node. If internetworking is not available then a dedicated node can also process the data to make decisions. Here different systems are developed using sensors and different hardware to make time sensitive decision makings. They are as follows.

5.2.1 Recognizing Real Time ECG Anomalies Using Arduino, AD8232 and Java

The functioning of Heart can be checked by continuous monitoring of heartbeats through Electrocardiogram (ECG). Irregularity in the rhythm of the heartbeat results in arrhythmia. Arrhythmia can be classified based on the origins that cause it. ECG signal comprises of PQRST wave. Analysis of PQRST wave helps identifying the type of arrhythmia. Thus, real-time analysis of ECG is of utmost priority, to acquire immediate medical aid and to avoid fatality. The paper discusses the use of the AD8232 sensor to capture ECG signals and its interfacing with Arduino Nano. Arduino is used as a Sampler and Analog to Digital Converter (ADC). The intervals of PQRST wave is analyzed using Java APIs and windowing algorithm. The results are compared with standard ECG signals to detect abnormalities and further analysis. The paper aids the reader to understand and develop a hand and low cost ECG analysis system, thus, reducing the treatment costs. In this section, authors have successfully proposed and implemented a system in Java that detects real-time anomalies in heartbeat. This system can be used by medical practitioners in both real-time and as well as in static modes. While both the modes require sampling frequencies, the real-time mode also makes use of a port connected to Arduino, a highly effective IoT unit. The AD8232 senor is then interfaced with Arduino. The final result displays the ECG signal on a Voltage-Time graph and also gives the

R-R, P-R, Q-R-S and Q-T intervals. The practitioners can easily and very accurately detect anomalies from the output.

5.2.2 Real-time Location tracker for critical health patient using Arduino, GPS Neo6m and GSM Sim800L in Health Care

In the health care sector, in case of an emergency, it is very crucial to know the exact location of the patient so that different critical health care services can be made available at the right time and place. This problem can be solved by using GPS coordinates. In this, an IoT device is made which locates the exact GPS coordinates of the patients to the server. Moreover, using the web interface on the server and Google Maps, doctors and hospital staff can track the exact location of the patient and serve him. IoT is the technology that can have control over any system present at remote locations via the internet. IoT devices are smaller, portable and available in different shapes, sizes, and capabilities. Multiple versions of Arduino can be used for different applications. For smaller and compact designing applications, Arduino nano is the best choice. The GPS Neo 6m gives accurate GPS coordinates and the refresh rate is also reasonable to update the changing locations. GSM Sim800L serves the purpose of gateway for the internet. It should have the working SIM card in it with active data plans. The ThingSpeak cloud is used which is freely available for IoT developers. It takes up dynamic readings and the same is updated in the database too. To use the google API, one has to pay the nominal amount as per the usage. Different functions available in libraries like SoftwareSerial, TinyGPS++, and AltSoftSerial.h are used for implementing the GPS. Thus, by using all available sensors and technology the live GPS tracker system is built with minimum cost.

5.2.3 IoT based Eye Movement Guided Wheelchair driving control using AD8232 ECG Sensor

Each and every muscular movement in the body is induced by electrical signals. These electrical signals are in mV and they are very sensitive to noise factors like electrical gadgets placed nearby, different movements, earthing, etc. If such signals are traced carefully, they can be used to accomplish multiple tasks. Such signals are called Myographs. This paper proposes a new method for eye-movement tracking, using Arduino Nano along with AD8232, i.e. the ECG Sensor. Most of the devices for Eye Tracking need to be placed right on the eye which sometimes use Infrared Radiations which may be harmful to eyes. This proposed method captures the gaze direction by muscular contraction, also called myography. This is done by placing the electrode pads on the forehead and the ECG line graphs demonstrate the direction of gaze which can be understood using the convolution method. After the movement direction is decided based on convolution method, the values are sent and received from the IoT cloud. Thus, the wheelchair movement can be controlled by online and offline modes, making it more opportune to the patient. The goal of the system is to avail low-cost solutions to the needer.

5.2.4 Analyzing ECG waves in Fog Computing Environment using Raspberry Pi Cluster

IoT and Cloud computing technologies are together serving different Health Care applications. These applications suffer from slower decision making due to network delays, less availability of bandwidth, transmission delays, processing delays and data de-noising. To avoid this, Fog computing can be applied as a middle layer between the IoT and Cloud layers. Here, we understand Fog computing, its characteristics applications along with its advantages and disadvantages. Fog computing greatly reduces transmission delays, but Fog devices lag in computing capabilities due to their limited processing power. This can be solved by deploying multiple devices and synchronizing their computation to enable parallel execution. In this paper, a single Raspberry Pi is used as a Fog node and multiple such Raspberry Pis are deployed

in the cluster form. The Dispy Python framework is used to allow parallel processing. Scalability is easy to achieve with Dispy, and also it provides different features like automated node discovery, job distribution, processing function distribution, remote access and enhanced security. The system is proposed and implemented to test the hypothesis that it improves the real time computation in health care applications. The final results are compared with a traditional processing system and it is found that the Raspberry PI cluster and Dispy can enhance the computation performance in Health Care

Chapter 6: Conclusion

This work introduced a cloud-assisted smart fog gateway for delay-sensitive IoT-driven healthcare applications. It ensures the tolerable delay while providing the service to the healthcare applications, by applying the smart partitioning and allocation using a decision tree. The decision rules are based on the application context and resource availability in fog and cloud infrastructure. In the smart fog gateway, the proposed approach intelligently takes the decision to determine the corresponding data stream based on the application context. The proposed approach provides the service to the end-user promptly. Cloud Computing Based IoT architecture is delay-sensitive for Critical Health Care applications. So, the LAN based Fog computing Processing approach can be used to reduce the delay. Also, this technique helps to reduce the data burden on the Cloud. Moreover, Fog Computing should have memory, processing and computation capabilities and Fog nodes can be placed in either LAN or as a Gateway. Since Raspberry Pi has networking, memory, storage and computation abilities, it becomes a suitable option to use as a Fog node. We also discuss different Raspberry Pi based Fog installations. Furthermore, fog based health care systems are better than the Cloud-based health care system in terms of network bandwidth and response time, but it lags behind in computation power. The overall response by fog to find any abnormality in the ECG signal is given way before the Cloud does - which is very vital in health care scenarios to save patient's lives. Fog computing gives a better response when the number of patients is less and it is observed that the Fog computing response time is directly proportional to the number of patients. After a particular threshold value of the number of patients, fog computing will not perform better than Cloud computing. The transmission delay and the computation delay plays a major role in the Fog computing domain.

Fog computing is able to do this with reduced transmission delays but to get reduce computational delay Raspberry Pi cluster is suggested. Dispy is a good tool to use in Pi cluster to facilitate ease of deployment and scalability in distributed computing. To get good performance from the dispy system, the assigned sub-job size should be optimal. The master node iterations and overheads depend on the sub-job size, which can affect the system performance at greater levels. Every hardware and software parameter matters a lot in terms of computation. In this system, four parameters namely response time, CPU usage, number of cores and memory is used. Each parameter has its effect on computation. For the current system, the CPU usage and number of cores were having a good impact while response time and memory had less impact on system performance. By considering these effects and their level of impact, OptiFog algorithm is designed with respect to different priorities and factors. The impact factor is a good measure to determine the processing health of any node. OptiFog algorithm performs fairly well for the ECG health care data using a Raspberry Pi cluster. OptiFog algorithm is designed for the worst-case scenarios so that it always performs better for average and worst case. OptiFog algorithm will show its computation variations in a heterogeneous environment where it is able to decide and assign the job size for different nodes. Shown test cases are justifying the performance of the OptiFog algorithm as, if the number of nodes and job increases, then the algorithm performance will also gradually increase in comparison to dispy systems. Hence OptiFog algorithm is able to achieve better computations in the Heterogeneous Raspberry Pi clustering environment in Fog computing. The suggested lightweight security algorithm is capable to achieve good level of security across any devices in the IoT-Fog scenario with very less time and space complexity.

Research Publication

[1] Kanani P., Padole M. (2018) Recognizing Real Time ECG Anomalies Using Arduino, AD8232 and Java. In: Singh M., Gupta P., Tyagi V., Flusser J., Ören T. (eds) Advances in Computing and Data Sciences. ICACDS 2018. Communications in Computer and Information Science, vol 905. Springer, Singapore. [**Scopus indexed**]

[2] Pratik Kanani and Mamta Padole, "ECG Image Classification using Deep Learning Approach", Handbook of Research on Disease Prediction Through Data Analytics and Machine Learning, IGI Global, pp.- 343-357. DOI: 10.4018/978-1-7998-2742-9.ch016 [**NLM indexed**]

[3] Pratik Kanani and Mamta Padole, "Deep Learning to Detect Skin Cancer using Google Colab", International Journal of Engineering and Advanced Technology (IJEAT), Vol. 8, Isuue. 6, pp. 2176-2183. [**Scopus indexed, UGC Care Journal]**

[4] Pratik Kanani and Mamta Padole, "IoT based Eye Movement Guided Wheelchair driving control using AD8232 ECG Sensor", International Journal of Recent Technology and Engineering, Vol. 8, Issue. 4, pp. 5013-5017. [**Scopus indexed, UGC Care Journal**]

[5] Pratik Kanani and Mamta Padole, "ECG Heartbeat Arrhythmia Classification Using Time-Series Augmented Signals and Deep Learning Approach", Third International Conference on Computing and Network Communications (CoCoNet'19). Procedia Computer Science journal, vol. 171(2020), pp. 524-531. [**Scopus, Web of Science and Ei Compendex indexed**]

[6] Pratik Kanani and Mamta Padole, "Exploring and Optimizing the Fog Computing in Different Dimensions", Third International Conference on Computing and Network Communications (CoCoNet'19). Procedia Computer Science journal, vol. 171(2020), pp. 2694-2703. [**Scopus, Web of Science and Ei Compendex indexed**]

[7] Pratik Kanani and Mamta Padole, "Improving Pattern Matching performance in Genome sequences using Run Length Encoding in Distributed Raspberry Pi Clustering Environment", Third International Conference on Computing and Network Communications (CoCoNet'19). Procedia Computer Science journal, vol. 171(2020), pp. 1670-1679. [**Scopus, Web of Science and Ei Compendex indexed**]

[8] Pratik Kanani and Dr. Mamta Padole, "Real-time Location Tracker for Critical Health Patient using Arduino, GPS Neo6m and GSM Sim800L in Health Care," *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, 2020, pp. 242-249, doi: 10.1109/ICICCS48265.2020.9121128. [**Scopus indexed**]

[9] P. Kanani and M. Padole, "Analyzing ECG waves in Fog Computing Environment using Raspberry Pi Cluster," 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2020, pp. 1165-1172, doi: 10.1109/I-SMAC49090.2020.9243398. [**Scopus Indexed**]

[10] Pratik Kanani and Mamta Padole, "Implementing and Analyzing Health as a Service in Fog and Cloud Computing", The International Journal of Intelligent Engineering and Systems, Vol. 13, No. 6, 2020. DOI: 10.22266/ijies2020.1213.13. [**Scopus Indexed and UGC Care Journal**]

[11] Pratik Kanani and Mamta Padole, "OptiFog: Optimization of Heterogeneous Fog Computing for QoS in Health Care", Journal of Theoretical and Applied Information

Technology, Vol. 98, No. 22, pp-3625-3642, November 2020. [**Scopus Indexed and UGC Care Journal**]

[12] Pratik Kanani and Dr. Mamta Padole, **"**An Effort to reduce the CO_2 emission in Computation for Green Computation**",** International Conference on Computing Technologies for transforming the Automated World-2020. (In Process of Publication) (**UGC Approved Journal**)

[13] **International Patent:** Mamta Padole and Pratik Kanani, "A SYSTEM FOR REAL-TIME HEART HEALTH MONITORING", Patent Number: 2020101730. [**Patent Granted**]

[14] **Indian Copyright:** "FOG OPTIMIZATION TECHNIQUE FOR QOS IN HETEROGENEOUS CLUSTERING ENVIRONMENT". [**Copyright Granted**]

[15] Pratik Kannai and Dr. Mamta Padole, "Light weight Multi-Level authentication scheme for secured Data transmission in Fog-IoT context", **[Copyright filed]**

References

[1] Atzori, L., Iera, A., and Morabito G, "The internet of things: A survey", Elsevier Computer networks, Vol.54, No.15, pp.2787-2805, 2010

[2] Botta, A., De Donato, W., Persico, V., and Pescapé, A, "Integration of cloud computing and internet of things: a survey", Elsevier Future Generation Computer Systems, Vol.56, pp.684-700, 2016

[3] Hassanalieragh, M., Page, A., Soyata, T., Sharma, G., Aktas, M., Mateos, G., and Andreescu S, "Health monitoring and management using Internet-of-Things (IoT) sensing with cloud-based processing: Opportunities and challenges", IEEE International Conference on Services Computing (SCC), pp.285-292, 2015

[4] Bonomi, F., Milito, R., Zhu, J., and Addepalli S, "Fog computing and its role in the internet of things", ACM Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp.13-16, 2012

[5] Chang, H., Hari, A., Mukherjee, S., and Lakshman, T. V, "Bringing the cloud to the edge", IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp.346-351, 2014

[6] Shi, Y., Ding, G., Wang, H., Roman, H. E., and Lu, S, "The fog computing service for healthcare", IEEE 2nd International Symposium on Future Information and Communication Technologies for Ubiquitous HealthCare (Ubi-HealthTech), pp.1-5, 2015

[7] Islam, S. R., Kwak, D., Kabir, M. H., Hossain, M., and Kwak, K. S, "The internet of things for health care: a comprehensive survey", IEEE Access, Vol.3, pp.678-708, 2015

[8] Bui, N., and Zorzi M, "Health care applications: a solution based on the internet of things", ACM Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies, p.131, 2011

[9] Doukas, C., and Maglogiannis I, "Bringing IoT and cloud computing towards pervasive healthcare", IEEE Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp.922-926, 2012

[10] Yi, S., Li, C., and Li, Q, "A survey of fog computing: concepts, applications and issues", ACM Proceedings of the Workshop on Mobile Big Data, pp.37-42, 2015

[11] Malik, S., Huet, F., and Caromel D, "Latency based group discovery algorithm for network aware cloud scheduling", Elsevier Future Generation Computer Systems, Vol.31, pp.28-39, 2014

[12] Ray, P. P, "Internet of things based physical activity monitoring (PAMIoT): an architectural framework to monitor human physical activity", IEEE Proceeding of CALCON, pp.32-34, 2014

[13] Li, F., Vögler, M., Claeßens, M., and Dustdar, S, "Efficient and scalable IoT service delivery on cloud", IEEE Sixth International Conference on Cloud Computing (CLOUD), pp.740-747, 2013

[14] Chiang, M., and Zhang T, "Fog and IoT: An overview of research opportunities", IEEE Internet of Things Journal, Vol.3, No.6, pp.854-864, 2016

[15] M. Aazam, and E. N. Huh, "Dynamiac resource provisioning through fog micro datacenter", In Proceedings of the 12th IEEE International Workshop on Managing Ubiquitous Communication and Services (MUCS '15), pp.105–110, 2015

[16] Xu, B., Da Xu, L., Cai, H., Xie, C., Hu, J., and Bu F, "Ubiquitous data accessing method in IoT-based information system for emergency medical services", IEEE Transactions on Industrial Informatics, Vol.10, No.2, pp.1578-1586, 2014

[17] Andriopoulou, F., Dagiuklas, T., and Orphanoudakis T, "Integrating IoT and Fog Computing for Healthcare Service Delivery", Springer Components and Services for IoT Platforms, pp.213-232, 2017

[18] Aazam, M., and Huh E. N, "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT", IEEE 29th International Conference on Advanced Information Networking and Applications (AINA), pp.687-694, 2015

[19] Aazam, M., and Huh, E. N, "Fog computing and smart gateway based communication for cloud of things", IEEE International Conference on Future Internet of Things and Cloud (FiCloud), pp.464-470, 2014

[20] Gia, T. N., Jiang, M., Rahmani, A. M., Westerlund, T., Liljeberg, P., and Tenhunen, H, "Fog computing in healthcare internet of things: A case study on ecg feature extraction", IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), pp.356-363, 2015

[21] Rahmani, A. M., Thanigaivelan, N. K., Gia, T. N., Granados, J., Negash, B., Liljeberg, P., and Tenhunen H, "Smart e-health gateway: Bringing intelligence to internet-of-things based ubiquitous healthcare systems", 12th Annual IEEEConsumer Communications and Networking Conference (CCNC), pp.826-834, 2015

[22] Gupta, H., Dastjerdi, A. V., Ghosh, S. K., and Buyya, R, "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things", Edge and Fog Computing Environments, arXiv preprint arXiv:1606.02007, 2016

[23] Alsaffar, A. A., Pham, H. P., Hong, C. S., Huh, E. N., and Aazam, M, “An Architecture of IoT Service Delegation and Resource Allocation Based on Collaboration between Fog and Cloud Computing”, Hindawi Mobile Information Systems, pp.1-15, 2016
[24] Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., and Liljeberg, P, “Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach”, Elsevier Future Generation Computer Systems, 2017
[25] Chakraborty, S., Bhowmick, S., Talaga, P., and Agrawal, D. P, “Fog Networks in Healthcare Application”, IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), pp.386-387, 2016
[26] Cao, Y., Hou, P., Brown, D., Wang, J., and Chen S, “Distributed analytics and edge intelligence: Pervasive health monitoring at the era of fog computing”, ACM Proceedings of the Workshop on Mobile Big Data, pp.43-48, 2015
[27] Dr Ibrahim samaha (2017, May 09). Simple Cardiology [Online]. Available: http://simple-cardio.blogspot.in/2013/01/importance-of-ecg-ekg.html
[28] Healthline. (2017, May 09). Meningitis [Online]. Available: http://www.healthline.com/health/meningitis#types2
[29] Msdmanuals. (2017, May 09). Meningitis [Online]. Available: http://www.msdmanuals.com/home/brain,-spinal-cord,-and-nerve-disorders/meningitis/acute-bacterial-meningitis
[30] Kanani P., Padole M. (2018) Recognizing Real Time ECG Anomalies Using Arduino, AD8232 and Java. In: Singh M., Gupta P., Tyagi V., Flusser J., Ören T. (eds) Advances in Computing and Data Sciences. ICACDS 2018. Communications in Computer and Information Science, vol 905. Springer, Singapore
[31] Muhammadd U. Bilal Ahmed B et al., “ Electrogram Feature Extraction and Pattern Recognition Using a Novel windowing Algorithm”, Advances in Bioscience and Biotechnology, 5, 886-894, October 2014.
[32] Computer - Memory, https://www.tutorialspoint.com/computer_fundamentals/computer_memory
[33] How to calculate a memory usage of a java program?,https://stackoverflow.com/questions/37916136/how-to-calculate-memory-usage-of-a-java-program
[34] Cloud Computing Saves Energy and CO_2 Emissions, http://www.energydigital.com/sustainability/cloud-computing-saves-energy-and-co2-emissions
[35] How is cloud influencing world data traffic?,https://www.ibm.com/blogs/cloud-computing/2013/04/how-is-cloud-influencing-world-data-traffic/
[36] The Megawatts behind Your Megabytes: Going from Data-Center to Desktop, http://aceee.org/files/proceedings/2012/data/papers/0193-000409.pdf
[37] With Internet Of Things And Big Data, 92% Of Everything We Do Will Be In The Cloud, https://www.forbes.com/sites/joemckendrick/2016/11/13/with-internet-of-things-and-big-data-92-of-everything-we-do-will-be-in-the-cloud/#553568ed4ed5
[38] Current Millis, https://currentmillis.com/
[39] D. Comer and D.L. Stevens. Internetworking with TCP/IP: Principles, protocols, and architecture. Internetworking with TCP/IP. Pearson Prentice Hall, 2006. ISBN: 9780131876712. url: https://books.google.co.in/books?id=jonyuTASbWAC.
[40] WHAT CAUSES THE INTERNET TO SLOW DOWN? https://www.colocationamerica.com/data-center-connectivity/speed-test.htm
[41] A. Faggiani, E. Gregori, A. Improta, L. Lenzini, V. Luconi and L. Sani, "A study on traceroute potentiality in revealing the Internet AS-level topology," 2014 IFIP Networking Conference, Trondheim, 2014, pp. 1-9.
[42] S. Branigan, H. Burch, B. Cheswick and F. Wojcik, "What can you do with Traceroute?," in IEEE Internet Computing, vol. 5, no. 5, pp. 96-, Sept.-Oct. 2001.
[43] Gareth Mitchell. “The Raspberry Pi single-board computer will revolutionise computer science teaching [For & Against]”. In: Engineering & Technology 7.3 (2012), pp. 26-26.
[44] Andrew K Dennis. Raspberry Pi home automation with Arduino. Packt Publishing Ltd, 2015.
[45] CG Raji et al. “Implementation of Bitcoin Mining using Raspberry Pi”. In: 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT). IEEE. 2019, pp. 1087-1092.
[46] Suzanne J Matthews et al. “Portable parallel computing with the raspberry pi”. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education. 2018, pp. 92-97.
[47] C. Pahl et al. “A Container-Based Edge Cloud PaaS Architecture Based on Raspberry Pi Clusters”. In: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (Fi-CloudW). 2016, pp. 117-124.
[48] P. Jutadhamakorn et al. “A scalable and low-cost MQTT broker clustering system”. In: 2017 2nd International Conference on Information Technology (INCIT). 2017, pp. 1-5.
[49] Pekka Abrahamsson et al. “Affordable and energy-efficient cloud computing clusters: The bolzano raspberry pi cloud cluster experiment”. In: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science. Vol. 2. IEEE. 2013, pp. 170-175.

[50] D. Borthakur et al. “Smart fog: Fog computing framework for unsupervised clustering analytics in wearable Internet of Things”. In: 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP). 2017, pp. 472-476.
[51] Richard Brown et al. “Teaching Parallel and Distributed Computing with MPI on Raspberry Pi Clusters: (Abstract Only)”. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education. SIGCSE '18. Baltimore, Maryland, USA: Association for Computing Machinery, 2018, p. 1054. ISBN: 9781450351034. doi:10.1145/3159450.3162369. url: https://doi.org/10.1145/ 3159450.3162369.
[52] G. L. Stavrinides and H. D. Karatza, "Task Group Scheduling in Distributed Systems," 2018 International Conference on Computer, Information and Telecommunication Systems (CITS), Colmar, 2018, pp. 1-5.
[53] A. Guermouche and J. - L'Excellent, "Memory-based scheduling for a parallel multifrontal solver," 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings, Santa Fe, NM, USA, 2004, pp. 71
[54] Xiaodong Zhang, Yanxia Qu and Li Xiao, "Improving distributed workload performance by sharing both CPU and memory resources," Proceedings 20th IEEE International Conference on Distributed Computing Systems, Taipei, Taiwan, 2000, pp. 233-241.
[55] L. Shi, Y. Sun and L. Wei, "Effect of Scheduling Discipline on CPU-MEM Load Sharing System," Sixth International Conference on Grid and Cooperative Computing (GCC 2007), Los Alamitos, CA, 2007, pp. 242-249.
[56] Kizhakkethil, Sree and S., Murugan. (2017). Memory based Hybrid Dragonfly Algorithm for Numerical Optimization Problems. Expert Systems with Applications. 83. 10.1016/j.eswa.2017.04.033.
[57] Mohammad I. Daoud and Nawwaf Kharma, " A hybrid heuristic–genetic algorithm for task scheduling in heterogeneous processor networks", Journal of Parallel and Distributed Computing, Volume 71, Issue 11, November 2011, Pages 1518-1531.
[58] H. Topcuoglu, S. Hariri and Min-You Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," in IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 3, pp. 260-274, March 2002.
[59] Dongning Liang, Pei-Jung Ho, Bao Liu. Scheduling in Distributed Systems.
https://cseweb.ucsd.edu/classes/sp99/cse221/projects/Scheduling.pdf
[60] Arash Ghorbannia Delavar,Mahdi Javanmard , Mehrdad Barzegar Shabestari and Marjan Khosravi Talebi, “RSDC (RELIABLE SCHEDULING DISTRIBUTED IN CLOUD COMPUTING)”, International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.2, No.3, June 2012.
[61] Luiz F. Bittencourt, Alfredo Goldman, Edmundo R.M. Madeira, Nelson L.S. da Fonseca, Rizos Sakellariou, "Scheduling in distributed systems: A cloud computing perspective", Computer Science Review 30 (2018) 31–54.
[62] Zafeirios C Papazachos, Helen D Karatza, "Gang scheduling in multi-core clusters implementing migrations", Future Generation Computer SystemsVol. 27, No. 8.
[63] Salim Bitam, SheraliZeadally and AbdelhamidMellouk (2018) Fog computing job scheduling optimization based on bees swarm, Enterprise Information Systems, 12:4, 373-397, DOI: 10.1080/17517575.2017.1304579
[64] F. A. Kraemer, A. E. Braten, N. Tamkittikhun and D. Palma, "Fog Computing in Healthcare–A Review and Discussion," in IEEE Access, vol. 5, pp. 9206-9222, 2017.
[65] D. R. Ries and G. C. Smith, "Nested Transactions in Distributed Systems," in IEEE Transactions on Software Engineering, vol. SE-8, no. 3, pp. 167-172, May 1982.
[66] dispy: Distributed and Parallel Computing with/for Python by GiridharPemmasani, https://pgiri.github.io/dispy/
[67] Raspberry Pi 3 Model B+, https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
[68] Raspberry Pi 4 Model-B with 4 GB RAM, https://robu.in/product/raspberry-pi-4-model-b-with-4-gb-ram/
[69] How to Impose High CPU Load and Stress Test on Linux Using ‘Stress-ng’ Tool, https://www.tecmint.com/linux-cpu-load-stress-test-with-stress-ng-tool/
[70] Kanani P., Padole M. (2018) Recognizing Real Time ECG Anomalies Using Arduino, AD8232 and Java. In: Singh M., Gupta P., Tyagi V., Flusser J., Ören T. (eds) Advances in Computing and Data Sciences. ICACDS 2018. Communications in Computer and Information Science, vol 905. Springer, Singapore
[71] Cardiology Teaching Package. http://www.nottingham.ac.uk/nursing/practice/resources/ cardiology/function/normal_duration.php
[72] Standard range of intervals, June 2017. E MEDICINE. http://emedicine.medscape.com/ article/2172196-overview
[73] Normal ECG. https://meds.queensu.ca/central/assets/modules/ECG/normal_ecg.html
[74] Eduardo Jose da S. Luz et al., "ECG-based heartbeat classification for arrhythmia detection: A survey", Computer Methods and Programs in Biomedicine, Volume 127, April 2016, Pages 144-164.

[75] Umer, Muhammad & Bhatti, Bilal & Tariq, Muhammad & Zia-ul-Hassan, Muhammad & Khan, Muhammad & Zaidi, Tahir. (2014). Electrocardiogram Feature Extraction and Pattern Recognition Using a Novel Windowing Algorithm. Advances in Bioscience and Biotechnology. 05. 886-894. 10.4236/abb.2014.511103.
[76] S. Sarkar, S. Chatterjee and S. Misra, "Assessment of the Suitability of Fog Computing in the Context of Internet of Things," in IEEE Transactions on Cloud Computing, vol. 6, no. 1, pp. 46-59, Jan.-March 2018.
[77] Muhammad Iqbal et al.,"Evaluating TCP performance of routing protocols for traffic exchange in street-parked vehicles based fog computing infrastructure", Journal of Cloud Computing: Advances, Systems and Applications (2020) 9:18 https://doi.org/10.1186/s13677-020-00159-w
[78] Sumit Singh Dhanda, Brahmjit Singh, Poonam Jindal, Lightweight Cryptography: A Solution to Secure IoT, Springer, 2020, Wireless Personal Communications, 112(4):1-34, June 2020